The practice of automatically building your code periodically is called continuous integration, the

resulting binary is after building the code can also be deployed automatically to the target environment.

This practice is called continuous delivery or continuous deployment.

as soon as new code is checked into the source code repository, it can be automatically built and tests can be run to validate

the new code.

This process not only validates the new code itself, but also ensures that it works in an integrated

manner with the rest of the code base.

**Jenkins** is one of the most popular tools for implementing continuous integration, continuous delivery

and continuous deployment pipelines for any platform or technology.

It is an open source automation tool written in Java with plug ins built for dev ops purposes.

Jenkins can trigger a build for every change made in the source code repository, for example, a git

repository.

Once the code is built, Jenkins deploys it on a test server for testing.

The concerned teams are constantly notified about the build and test results.

Finally, Jenkins deploys the build application on the production server.

Thus, Jenkins helps in cutting down costs by enabling teams to automate their build and release process.

To enhance its power and usability, Jenkins supports **thousands of plug ins.**

There is a huge and active community behind Jenkins and they are constantly contributing to and updating

the plugin ecosystem, which makes the Jenkins platform even more powerful and flexible every day.

In an improbable scenario where you cannot find a suitable plug in for your project or a situation,

you yourself can extend Jenkins by developing additional plug ins.

That is the advantage of using an open source platform which has a strong and **generous community** behind

it.

The key tangible benefits of Jenkins are that it increases the **developer productivity**

and brings greater agility to the development process by automating the mundane but important tasks.

**SUMMARY:**

To summarize, Jenkins is an open source platform that can handle any kind of build, continuous integration,

continuous delivery or dev ops task, you can integrate Jenkins with a number of testing and deployment

technologies.

Jenkins is a powerful platform that allows continuous integration and continuous delivery of projects,

regardless of the platform you are working on.

Jenkins can be extended by additional plugins.

It is a tried and tested platform which is widely used in the industry, from small startups to very

large corporations.

**How does Jenkins work?**

Well, there is no surprise that Jenkins implements different devops steps with the help of plug ins, plugins allow the integration of various devops stages within the Jenkins framework.

If you want to integrate a particular tool stack framework or platform, you need to install the jenkins' plug ininto that particular tool or framework so that Jenkins' can work with it seamlessly.

For instance, you may want to use git via git plugin Mavin to via Mavin plugin and some cloud related

frameworks using A.W. as are as are specific plugins.

**Architecture of Jenkins?Core components?**

Jenkins uses the master slave architecture to manage distributed builds

Master and slave communicate through TCP IP protocol.

The main jenkins' server in this distributed architecture is the master server.

The master job is to   
1) pull the system repositories, which could be anything from git to subversion to cvs, etc..

2) Schedule build Jobs   
3) dispatch builds to the slaves for the actual execution,   
4) monitor the slaves, possibly taking them online and offline as required

5) record and present the build results.

So primarily, the master's job is to orchestrate the builds and jobs across the whole cluster.

That being said, a master's instance of Jenkins' can also execute build jobs directly, whereas a slave jenkins' instance is a Java executable that runs on a remote machine,   
the slave jenkins' instances accept request from the Jenkins' Master instance and execute the job and report back to the status of the master.

Slaves can run on a variety of operating systems.

You can configure a project to always run on a particular slave machine or on a certain type of slave machine, or simply let Jenkins pick the next available slave.

For example, in this diagram you can see that Jenkins' architecture consists of a jenkins' master which is managing two jenkins' slaves.

The Jenkins' master distributes the work load to the slaves, which accepts the request given by the master.

Additional slave nodes can be added to this architecture as the load on the system grows.

**SUMMARY:**

dev ops practices like continuous

integration, continuous delivery and continuous deployment help in increasing the productivity and

streamlining the software development process.

Developed practices reduce the overall risk of the project by discovering errors early in the cycle

and thus increases the overall quality of the product.

Jenkins is an excellent opensource automation tool written in Java with plug in support.

It is a tried and tested platform for implementing our dev ops pipelines and components.

**INSTALLING JENKINS PREREQUISITES**

install Jenkins on your **Linux instance**, you will need to ensure that **Java has been**

**installed** on your Linux instance because this is a necessary prerequisite for installing and running

Jenkins'.

**On a linux machine install JDK and set JAVA\_HOME env variable**

Use this command to see the path which needs to be set **Update-alternatives –display java**

When we navigate to our artifactory instance, we are basically accessing a Web application which has

been deployed into Apache Tomcat server Instance. Arti factory is just a Java Web application which has been deployed as a war file into Apache

Tomcat.

What is the significance of this?

It means that after installation of artfactory you will have a full installation of Apache Tomcat on

your server.

And this server is not exclusive to artfactory only, you can just as easily install other Tomcat Web

apps and Tomcat admin Web apps by executing the youm install command into this instance of Apache Tomcat.

This is convenient because if you are working with ATA factory, there is a good chance you will be

working with other Java based Web applications so you can unify the applications.

And your artfactory instance into the same Apache Tomcat server

It's centralizes the management of all the binary artifacts generated and used by the organization to

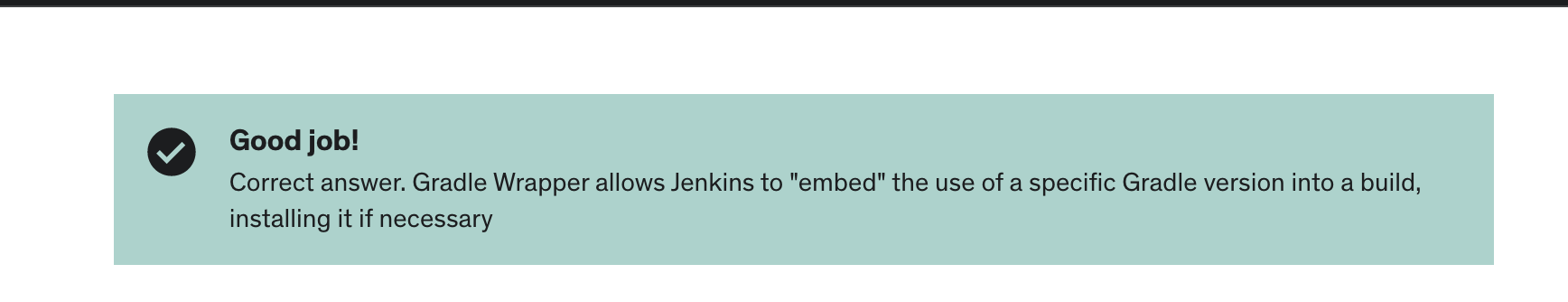
overcome the complexity arising from the diversity of binary artifact types, their position and the

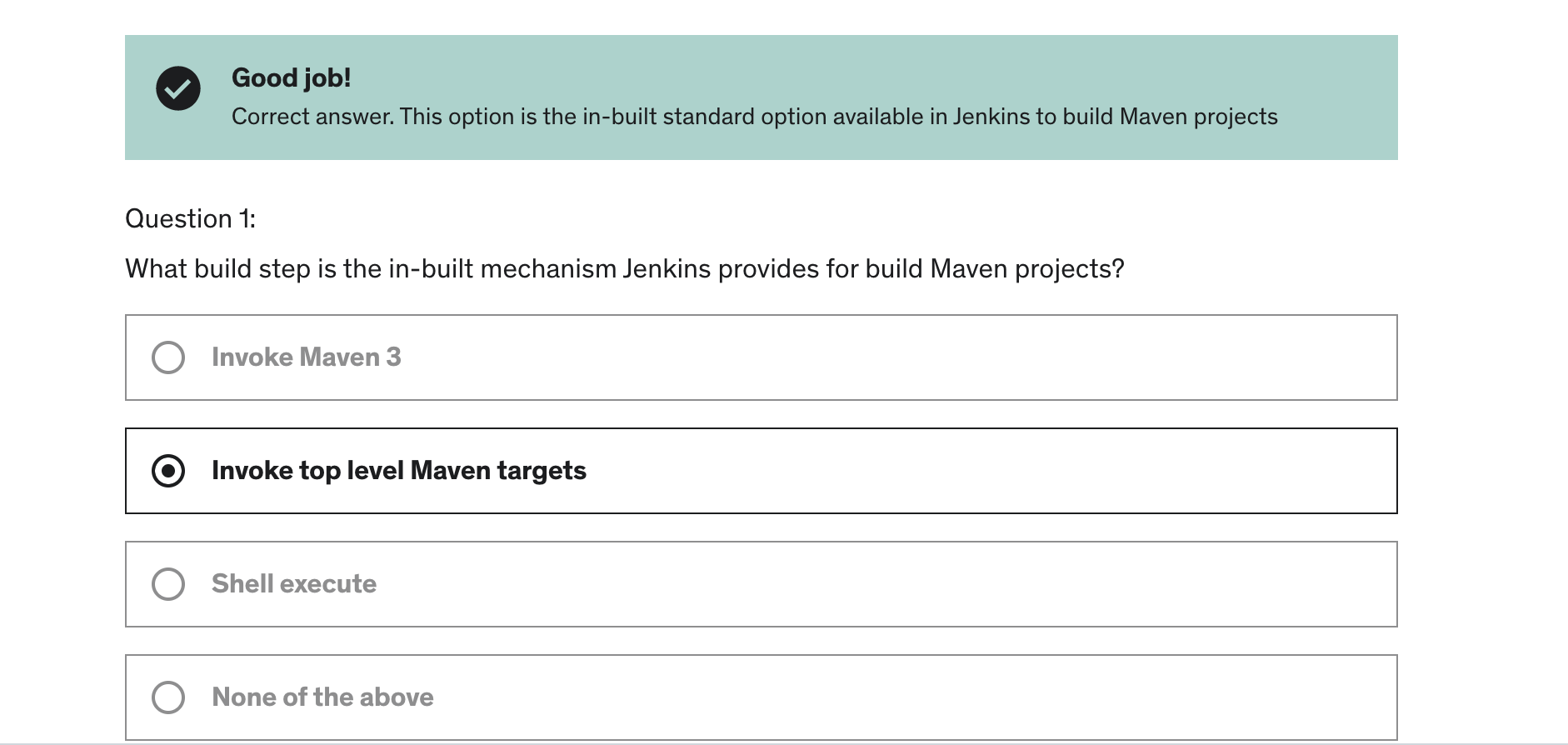
overall workflow and the dependencies between them.

Artifactory integrates with all major CICD and dev ops tools like Jenkins', Team, City, TFS, etc.,

and it provides an end to end automated and reliable solution for tracking artifacts from development

to production.





SQITCH:

The idea here is that we will submit all our schema and source code changes from our database to our

git repository, then we will create a jenkins' job to pull the latest source from our database git

repository and update our database with the changes in a consistent manner.

managing source code for database

changes and applying them consistently across multiple environments.

It's very hard to do.

For managing the schema of a relational databases in a consistent manner, we will use a popular versioning

and source control management tool called SQITCH.

Skitch is a database change management framework which can help us manage our database changes in a

consistent manner, it supports scenarios where multiple developers on the development team are checking

in Changes simultaneously in an agile fashion .Sqitch keeps track of all these changes and is able to apply

them to target database environments.

So when any team works on the same database at the same time, they just need to add

their commits to the sqitch repository, using the sqitch commands sqitch will ensure everything is

kept track of and managed well.

In fact, many of the sqitch commands are very similar to git.

What is workspace in Jenkins: for each job on Jenkins,Jenkins allocates a unique workspace directory.

This is the directory where the code is checked out and builds activities take place.

In most cases, we should let Jenkins' manage this directory and its contents.

But as we have seen in various job definitions that we created, it is a good practice to select an

option to delete this workspace directory before a new build is triggered.

This will ensure that the current build is clean and the files and artifacts from the previous build

do not interfere with the current build.

**Jenkins Security settings:**

Manage JENKINS -> CONFIGURE GLOBAL SECURITY.

One of the first options we see is to be able to configure our TCP port for JNLP agents.

This is the TCP port on which our Master Jenkins server will allow the slaves to connect via Java based

technology called JNLP or Java Network Launch Protocol. And it enables an application to be launched on a client by using resources that are hosted on a remote

Web server. These JNLP port settings play a critical role when connecting slaves to the Master Jenkins server. By default, this port is disabled, but to enable it, we have two options, we can either set it to

a fixed port or a random port .Under fixed option.

We will need to provide a specific port for these connections.

This option may be better if you were behind a complex corporate network and you have to configure firewalls

between master and slave nodes.

But for usual setups, we're all master and slave nodes are in the same network.

Random port is recommended here.Jenkins' will pick an available port automatically.

Next, we come to the access control section, this section controls the core of the security and permissions

settings.

When users log on to Jenkins' UI, they are authenticated and an appropriate set of permissions are

assigned to them based on these settings, the security setting within Jenkins' is controlled mainly

by two axis security realm, which determines users and their passwords, as well as what groups the

users belong to.

Authorisation strategy, which determines who has access to what these two axes are orthogonal and need

to be individually configured.

the Master Jenkins server is the central controller

node, which coordinates and assigns build jobs across the cluster of nodes.

So it is very important that we keep the Master Jenkins node very responsive, especially since most

people accessing Jenkins will do so via the Web interface, which is running on the master node.

Try to remember that running and managing build jobs is a demanding task and requires a lot of memory

and CPU resources, so it is advisable not to run any jobs on the master node.

If you were involved in large projects and if you choose to run your jobs on the master node, you may

run into situations where the master note is overwhelmed, running the build jobs and has no resources

left for running the web interface or managing other aspects of the jenkins' cluster.

Cluster can become unresponsive

Furthermore, to protect against these disastrous scenarios, don't forget to make regular backup configurations

for your jenkins' master nodes configuration.

Now moving on to the slave nodes.

First of all, the slave node should be quick to set up and throw away if they suffer any type of software

or hardware issues.

Plan your slave server for high usage and optimal efficiency.

It should be utilized to the maximum capacity and you should run multiple jobs on it, not just a single

job.

Always plan the slave nodes hardware for really high usage.

Regularly and diligently monitor your CPU and disk usage over the slave nodes.

It is also a good idea to segregate heavy and more complex jobs to a separate set of beefy servers so

they do not end up blocking the smaller and quicker jobs in the queue.

In case any of these jobs are too demanding, it is even advisable to isolate them and assign them to

a dedicated slave nodes so that they do not end up affecting the efficiency of the entire build queue.

That is how you avoid all problems with accidentally failed jobs.

Another good practice is to pick your plugins for jenkins' responsibly.

Once installed, please ensure they are configured properly and you regularly update their versions.

Moreover, please also take care to clean up unnecessary plug ins.

These unnecessary plug ins may end up cluttering the whole system and may cause unforeseen issues.

Another good practice is to install and configure logging and monitoring for your jenkins' nodes themselves

so that you can easily and quickly troubleshoot the problems when they arise.

Another good practice about build jobs configuration is to keep the logic for the build.

Jobs, simple and straightforward, try to avoid too many steps in a single job.

If you do need multiple steps, it may sometimes be easier to break them into logical groups and configure

them as separate jobs which are chained together.

This achieves the same purpose, but the build configuration is much easier to maintain when they

are manageable size.

Avoid using any complex logic in your shell command steps, we use shell commands in our course because

most of the jobs needed very straightforward shell commands without any direct rules embedded in them.

As the complexity of the jobs grow, using too many shell steps just multiplies that complexity exponentially.

Another usual problem you may run into is source code management and its integration with Jenkins'.

It is always a good practice to manually pull the source code on the jenkins' node and verify whether

Jenkins' can access the source code directly or not, you can even take it one step further and run

the entire build manually on the node before automating it with Jenkins'.

What are some concerns about archiving your artifacts in jenkins'?

Jenkins is not optimal at this at all.

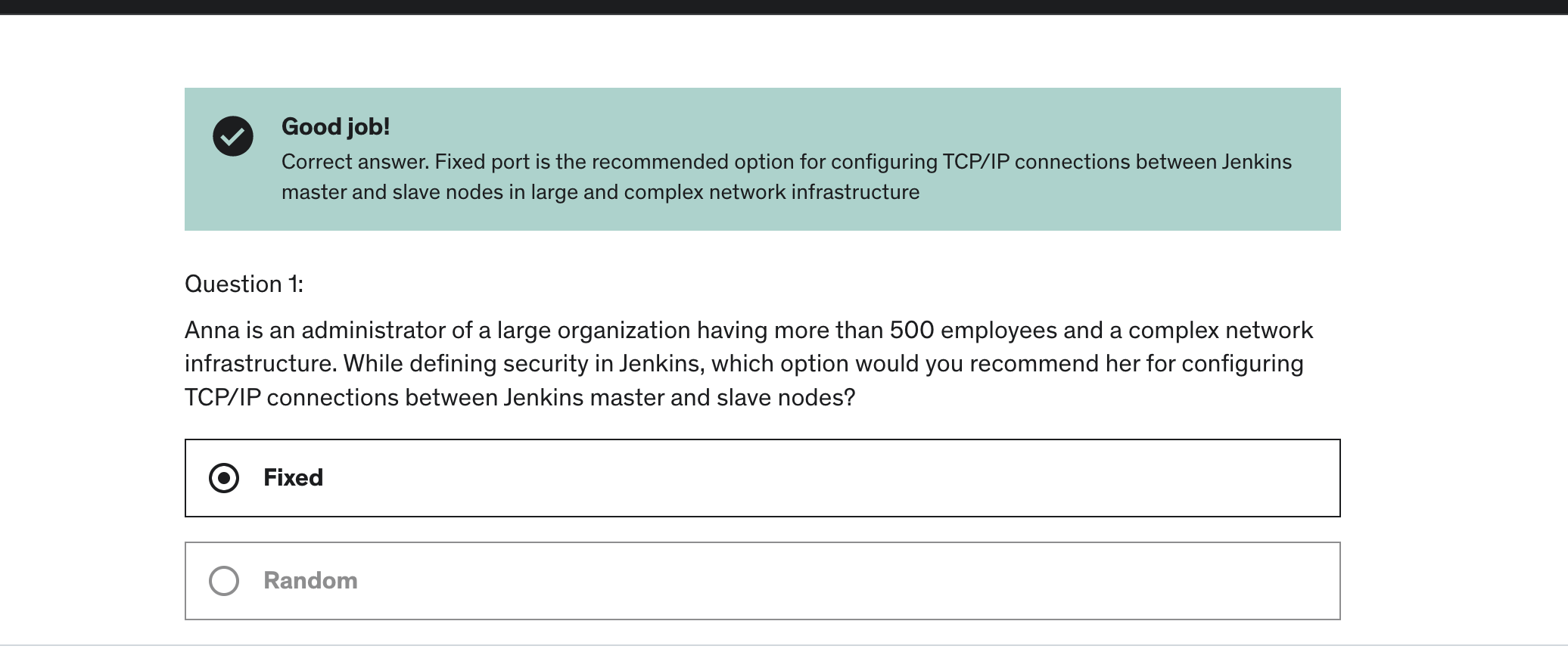
This is why we suggest you use a repository manager like artfFactory for storage and management of your

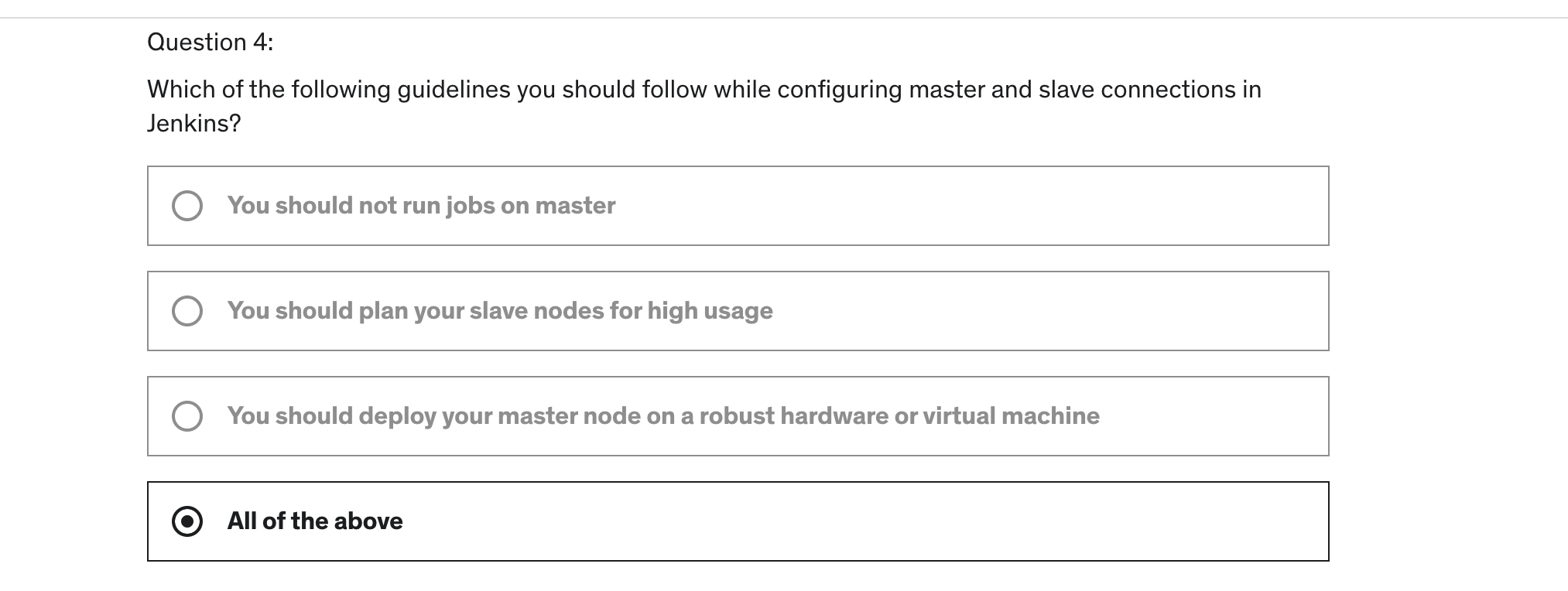
build artifacts.

The solutions are much more robust and easier to manage than Vanilla Jenkins.

Moreover, most of these repository manager solutions work really, really well with Jenkins', in most

cases with the help of plugins.





Number of executors setting in slave nodes: The maximum number of jenkins' jobs is dependent upon what you set as the limits in the master and slaves.

Usually we specify this limit based on the number of CPU cores in your slave machine, but your mileage

may vary depending upon available memory, disk, space, availability of SSD and overlap of source

code.

The limit is controlled by this setting “number of executors”