

[Home](#) > [Interview](#) > [Spring, Hibernate, & Maven Interview Q&A](#) > [Git & SVN](#) > 8 Git

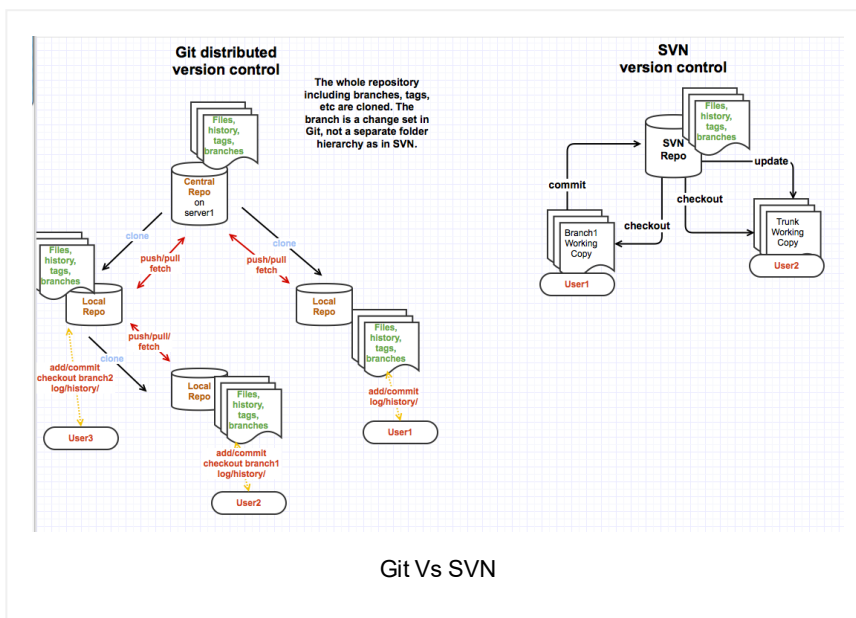
Source control system interview questions & answers

# 8 Git Source control system interview questions & answers

Posted on [March 24, 2015](#) by [Arulkumaran Kumaraswamipillai](#)

**Q1.** How does Git differ from SVN?

**A1.**



**600+ Full Stack Java/JEE Interview Q&As ♥Free ♦FAQs**

[open all](#) | [close all](#)

[Ice Breaker Interview](#)

[Core Java Interview C](#)

[JEE Interview Q&A \(3](#)

[Pressed for time? Jav](#)

[SQL, XML, UML, JSC](#)

[Hadoop & BigData In](#)

[Java Architecture Inte](#)

[Scala Interview Q&As](#)

[Spring, Hibernate, & I](#)

[Spring \(18\)](#)

[Hibernate \(13\)](#)

[AngularJS \(2\)](#)

[Git & SVN \(6\)](#)

[♥ Git & Maven fc](#)

[♥ Merging Vs rel](#)

[♥ Understanding](#)

[6 more Git interv](#)

[8 Git Source cor](#)

[Setting up Cygw](#)

[JMeter \(2\)](#)

[JSF \(2\)](#)

[Maven \(3\)](#)

**#1.** Git is a **distributed** source control system meaning that there will be multiple client repositories. SVN is one repository with lots of clients. GIT is decentralized to a point where people can track their own edits locally without having to push things to an external server.

### In SVN:

You checkout from a trunk or a branch from a central SVN repository with the **checkout** or “**co**” command.

### In Git:

**a)** You “**clone**” a company wide “master” repository to your local repository. When you clone, you are making a copy of the whole repository including the branches and the tags. After cloning the whole repository, you can switch to a particular branch or tag within your local repository with the “**checkout**” command.

**b)** Since the “clone” command copied the whole repository to your local machine, you can perform your own diff, logs, etc without requiring to remotely connect to your **origin** (e.g. master repository or where you cloned your local repository from). (i.e. where you cloned your local repository from).

**#2.** Git is a file content management tool made to merge files, evolved into a true Version Control System.

## How can Git copy the whole repository? Wouldn't it be too many files to copy?

### In SVN:

















Branches & tags are copy of a directory. So, you have many copies of same files in various folders.

### In Git:

-  [Testing & Profiling/Sa](#)
-  [Other Interview Q&A 1](#)
-  [Free Java Interview](#)

## 16 Technical Key Areas

[open all](#) | [close all](#)

-  [Best Practice \(6\)](#)
-  [Coding \(26\)](#)
-  [Concurrency \(6\)](#)
-  [Design Concepts \(7\)](#)
-  [Design Patterns \(11\)](#)
-  [Exception Handling \(3\)](#)
-  [Java Debugging \(21\)](#)
-  [Judging Experience I](#)
-  [Low Latency \(7\)](#)
-  [Memory Management](#)
-  [Performance \(13\)](#)
-  [QoS \(8\)](#)
-  [Scalability \(4\)](#)
-  [SDLC \(6\)](#)
-  [Security \(13\)](#)
-  [Transaction Managen](#)

## 80+ step by step Java Tutorials

[open all](#) | [close all](#)

-  [Setting up Tutorial \(6\)](#)
-  [Tutorial - Diagnosis \(2\)](#)
-  [Akka Tutorial \(9\)](#)
-  [Core Java Tutorials \(2\)](#)
-  [Hadoop & Spark Tuto](#)
-  [JEE Tutorials \(19\)](#)
-  [Scala Tutorials \(1\)](#)
-  [Spring & Hibernate Ti](#)
-  [Tools Tutorials \(19\)](#)

You work with changes, and not files. In Git, branches are part of the history of data (i.e. and not data itself), and where tags are a true meta-data. This is why you can clone the whole repository without having to worry too much about if you have enough space.

**Q2.** Does Git provide a better version control compared to SVN?

**A2.**

**#3.** SVN is designed to be a more centralized version control system whereas GIT is based on each user having his/her own GIT repository and those repositories push changes back up into a central one from time to time. In other words decentralized or distributed as shown above in the diagram. With your Git local repository cloned from the central repository, you can perform diffs, merges, log, etc for a better local version control. When you are happy with your changes, then you can share it with your team by “**push**” ing your changes to the master by making a remote call via **ssh** or **http**.

You can even clone the local repositories from the other users before those changes are pushed to the central repository. Git shines when you are decentralized and want to work not only at your regular job, but with your server at home and the laptop on the move you have the clone of the whole repository to work on. You can “push” (i.e. sync) your local repository back to your central repository when you have network connection. You can't do this with SVN, which is a centralized client/server paradigm.











SVN is a bit simpler to learn and understand as it has a few commands “via ssh or http” that you execute against the central repository from your SVN client. With Git, you have both local commands that you execute against your local repository, and remote ssh or http commands that you use to sync from your local repository with a remote one.

**#4.** Even though Git has been there for a while, it has really gained momentum after “**GitHub**” where many open-source

 [Other Tutorials \(45\)](#)

## 100+ Java pre-interview coding tests

[open all](#) | [close all](#)

-  [Can you write code? \(1\)](#)
-  [♦ Complete the given](#)
-  [Converting from A to I](#)
-  [Designing your classe](#)
-  [Java Data Structures](#)
-  [Passing the unit tests](#)
-  [What is wrong with th](#)
-  [Writing Code Home A](#)
-  [Written Test Core Jav](#)
-  [Written Test JEE \(1\)](#)

## How good are your .....?

[open all](#) | [close all](#)

-  [Career Making Know-](#)
-  [Job Hunting & Resum](#)

projects are stored. The flagship functionality of GitHub is “**forking**”, which is copying a repository from one GitHub user’s account to another. This enables you to take a project that you don’t have write access to, and use it for learning & modifying it under your own account. If you make changes you’d like to share, you can send a notification called a “pull request” to the original owner. That user can then, with a click of a button, merge the changes found in your repo with the original repo.

So, you can make an international level collaborative effort in building your own products and services with other developers in different parts of the world. You can show off your work to prospective employers, investors, and clients by having a “GitHub” account.

Both Git and SVN have their strengths and weaknesses, but knowing Git will be like having another powerful tool in your toolkit.

**Q3.** In Git, which commands make remote calls and which are local commands?

**A3.**

**Remote:** clone, push, pull, fetch, remove, etc.

**Local:** status, commit, checkout, merge, diff, log, add, reset, init, tag, revert, hist, mv, etc

**Q4.** In Git, how do you create a new repository?

**A4.**

**Step 1:** Create a project directory and a file.

```
1 mkdir myproject
2 cd myproject
3 touch hello.html
4
```

**Step 2:** Create a repository.

```
1 git init
```

This would have created a “.git” directory for meta data under “myproject”.

**Step 3:** Add the file “hello.html” to the repository

```
1 git add hello.html #staging
2 git commit -m "My first Commit" #commit
3
```

After staging, if you want to look at the status you can do

```
1 git status
```

If not happy with the changes, you can revert it with

```
1 git reset hello.html
2 git reset --hard #to reset all changes
3 git checkout HEAD -- file.txt #revert one file
4
```

**Step 4:** Checking the status of your repository as to if there are any changes to be committed.

```
1 git status
```

then,

**Step 5:** push it to the central repository if you have one.

```
1 git push origin myproject
```

to make further changes

**Step 6:** git pull to get the changes other users may have made

**Step 7:** make your changes in eclipse or other IDEs.

**Step 9:** add (i.e. staging or queuing the changes) & commit the changes to your local repository with comments.

**Step 10:** See the history. check the status.

```
1 git log
```

```
1 gitk --all
```

**Step 11:** push the changes to the central repository.

**Q5.** What if your organization already has a corporate wide **master repository**, how would you go about working with Git?

**A5.**

**Step 1:** Clone the master repository to your local. This brings the whole repository including the branches and the tags, which are actually changes or history,

```
1 git clone https://mycompany.com/myproject.git
```

or

```
1 git clone ssh://mycompany.com/path/to/myproject.g
```

**Note:** You can also clone local repositories by providing the path. So, you can have multiple local repositories.

**Step 2:** Your cloned local repository may have multiple branches. So, select which branch to be the current to work on. You do this with a “**checkout**” command.

```
1 git checkout mybranch
```

**Step 3:** You can use the status command to see what your current branch is. It will have an \* next to it. You can switch

branches within your cloned local repo with the “checkout” command.

**Step 4:** You can also list your tags with

```
1 git tag
```

or

```
1 git tag -l "1.0.*"
```

**Step 5:** Once you are happy with your committed changes, you can push them to the **origin**. In this case the corporate wide master repository is our origin. This could be your other users’ local repositories.

```
1 git push --all origin
```

**Note:** The **origin** being  
“ssh://mycompany.com/path/to/myproject.git”

**Step 6:** You can sync your local repository with the others’ changes.

```
1 git pull --all origin
```

**Note:** The “pull” command does two things **1) fetch** the changes from the origin **2) merge** the changes into your repository’s active branch. You can also do a fetch, followed by a status command to review the changes others have made, and then merge those changes into your local repository.

**Q6.** What tools do you use to work with Git? Are there any graphical tools?

**A6.** Git is very powerful with Unix like commands. In windows, you can use **Cygwin** or **msysGit**. There are

graphical tools like **SourceTree**, which is a graphical git client for windows. **gitk** is another GUI tool.

“git difftool” will launch “WinMerge” GUI for understanding the differences and merging.

**Atlassian Stash** is Git repository software, which enables collaboration on Git repositories. It provides enterprise level support and integration to other SDLC tools like JIRA (for defect management), Bamboo (for continuous integration & build), etc.

**Q7.** How do SVN and Git commands differ?

**A7.** Modifying an existing file SVN vs Git.

## SVN

- 1) make changes to a file.
- 2) svn commit # a ssh, file or http call

## Git

- 1) make changes to a file.
- 2) git add (i.e. staging) #local
- 3) git commit -m “changed x to y” #local
- 4) git push # (i.e. ssh to the central rep)

**Q8.** What is a **feature branch** in Git?

**A8.** Feature branches (or sometimes called topic branches) are used to develop new features for the upcoming releases.

The core idea behind the **feature branch** workflow is that all feature development should take place in a dedicated branch instead of the master branch. This encapsulation makes it easy for multiple developers to work on a particular feature without disturbing the main codebase. It also means the master branch will never contain broken code, which is a huge advantage for continuous integration environments.



The maintainers (e.g team leads, etc) of Git repository prefer the team members' changes to be on a feature branch, as they can be reviewed or signed off by the maintainer(s) before merged to the main codebase.

The "feature branch workflow" still uses a central repository, and master still represents the official project history. But, instead of committing directly on their local master branch, developers create a new branch every time they start work on a new feature. Feature branches should have descriptive names, like "floating menu feature" or JIRA-#999.

**1)** Clone from the origin. Origin can be a **remote** master repository.

```
1  
2 git clone master  
3
```

**2)** Switch to the feature branch once you have clone the whole repository to your **local** machine.

```
1  
2 git checkout -b JIRA-#999      # switched to a  
3
```

**3)** Make some changes to files within your feature branch.

**4)** Stage the changes to the feature branch in the local repository.

```
1  
2 git add                      #staging the cha  
3
```

**5)** Commit the changes to the feature branch in the local repository.

```
1  
2 git commit "new feature added"  
3
```

6) Push the changes to the remote repository's feature branch.

```
1  
2 git push -u origin JIRA-#999 #feature branch  
3
```

7) log into web based central repository managers like **Github** or **Atlassian Stash** to see your changes.

8) switch to “JIRA-#999” feature branch, and issue a “**pull request**” so that the maintainer can merge it. The pull request is more than just a notification to the fellow repository users of your changes. It's a dedicated forum for discussing the proposed feature & reviewing your code for any improvements. If there are any problems with your changes, your team mates can reject your pull request with comments. When your pull request gets approved by your team mates, the changes in the feature branch gets merged to the master. All of these pull request activities are tracked.

You need 4 pieces of information to file a pull request:

- a) the source repository,
- b) the source branch,
- c) the destination repository,
- d) and the destination branch.

## What is the key benefit of a “pull” request?

Once someone completes a feature, he/she doesn't immediately merge it into master. Instead, he/she pushes the feature branch to the central server and file a pull request asking the fellow developers to review & merge his/her additions into master. This gives other developers an opportunity to review the changes before they become a part of the main code base. The 3 key benefits of this approach are:

- 1) Improves code quality due to continuous code reviews.

- 2) Communicates changes promptly to the team mates.
- 3) Enforces better & collective code ownership. Do you want your fellow developers to comment on or take ownership of your sloppy code?

## Popular Member Posts

♦ 11 Spring boot interview questions & answers

904 views

♦ Q11-Q23: Top 50+ Core on Java OOP Interview Questions & Answers

816 views

001A: ♦ 7+ Java integration styles & patterns interview questions & answers

427 views

18 Java scenarios based interview Questions and Answers

408 views

♦ 7 Java debugging interview questions & answers

324 views

01b: ♦ 13 Spring basics Q8 – Q13 interview questions & answers

311 views

01: ♦ 15 Ice breaker questions asked 90% of the time in Java job interviews with hints

304 views

♦ 10 ERD (Entity-Relationship Diagrams) Interview Questions and Answers

301 views

♦ Q24-Q36: Top 50+ Core on Java classes, interfaces and generics interview questions & answers

251 views

♦ Object equals Vs == and pass by reference Vs value

234 views



Bio

Latest Posts



## Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)



### About Arulkumaran Kumaraswamipillai

Mechanical Eng to freelance Java developer in 3 yrs. Contracting since 2003, and attended 150+ Java job interviews, and often got 4 - 7 job offers to choose from. It pays to prepare. So, published Java interview Q&A books via [Amazon.com](https://www.amazon.com) in 2005, and sold 35,000+ copies. Books are outdated and replaced with this subscription based site. **945+** paid members. [join my LinkedIn Group](#). [Reviews](#)

◀ Working with a given input string & character processing

♥ 8 AngularJS interview questions & answers ▶

Posted in Git & SVN, member-paid

## Empowers you to open more doors, and fast-track

### Technical Know Hows

☀ [Java generics in no time](#) ☀ [Top 6 tips to transforming your thinking from OOP to FP](#) ☀ [How does a HashMap internally work? What is a hashing function?](#)

☀ [10+ Java String class interview Q&As](#) ☀ [Java auto un/boxing benefits & caveats](#) ☀ [Top 11 slacknesses that can come back and bite you as an experienced Java developer or architect](#)

### Non-Technical Know Hows

☀ [6 Aspects that can motivate you to fast-track your career & go places](#) ☀ [Are you reinventing yourself as a Java developer?](#) ☀ [8 tips to safeguard your Java career against offshoring](#) ☀ [My top 5 career mistakes](#)

## Prepare to succeed

☀ [Turn readers of your Java CV go from “Blah blah” to “Wow”?](#) ☀ [How to prepare for Java job interviews?](#) ☀ [16 Technical Key Areas](#) ☀ [How to choose from multiple Java job offers?](#)

Select Category ▼

## © Disclaimer

The contents in this Java-Success are copy righted. The author has the right to correct or enhance the current content without any prior notice.

These are general advice only, and one needs to take his/her own circumstances into consideration. The author will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. No guarantees are made regarding the accuracy or usefulness of content, though I do make an effort to be accurate. Links to external sites do not imply endorsement of the linked-to sites.