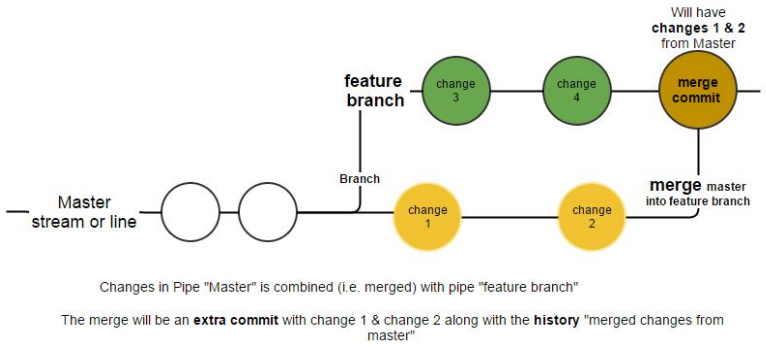
♥ Merging Vs rebasing on Git questions and answers

Posted on May 11, 2015 by Arulkumaran Kumaraswamipillai

Q1. What do you understand by the terms merging and rebasing? Can you explain both with a daiagram?

A1. Both of these commands are designed to integrate changes from one branch into another branch—they just do it in very different ways. When developers work in parallel and commit changes to different streams of the same code base, eventually some or all of these commits have to be brought together into a shared graph, and merging and rebasing are two primary ways that let us do that.

Merging brings two lines of development together while preserving the ancestry of each commit history. It is like melting two different pipes together. The pipe itself doesn't break, it's just combined with another pipe. So, the commit itself knows that it is a merge commit.



SCM: Merging

```
1 git checkout feature-branch
2 git merge master
3
```

or

```
1 git merge master feature-branch
```

In contrast, **rebasing** is like cutting off a pipe and welding it on another pipe. Rebasing unifies the lines of development by re-writing changes from the source branch so that they appear as children of the destination branch – effectively pretending that those commits were written on top of the destination branch all along.

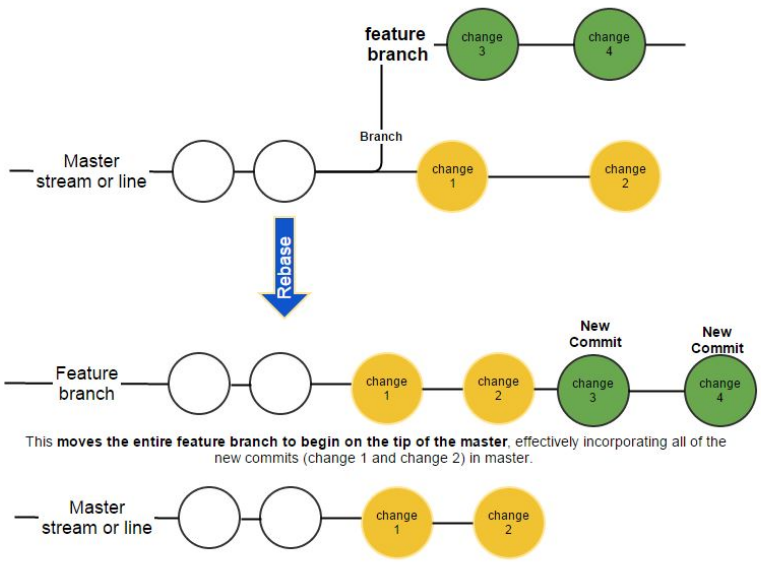
Testing & Profiling/Sa
Other Interview Q&A
Free Java Interviev

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 Managemen
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 Tu
Tools Tutorials (19)



SCM: Rebasing

```
1 git checkout feature-branch
2 git rebase master
```

Q2. What are the benefits of rebasing? When will you favor a merging over a rebasing?

A2. The major benefit of rebasing is that you get a much cleaner project history as

1) Rebasing eliminates the unnecessary “merge commits” required.
2) Rebasing also results in a perfectly linear project history that you can follow the tip of feature all the way to the beginning of the project without any forks. Frequent merging creates a fish bone like history that becomes very hard to read.

Having said this, you have to know **“when not to rebase”**.

The golden rule of git rebase is to never use it on public branches. In the above diagram, you rebased the “feature branch onto the master branch”. In other words, you took the changes from “master” into the feature-branch that you are working on.

⊞ Other Tutorials (45)

100+ Java pre-interview coding tests

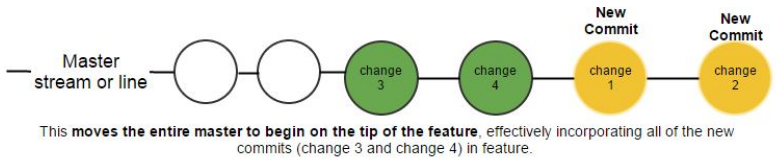
open all | close all
⊞ Can you write code? (
⊞ ♦ Complete the given
⊞ Converting from A to I
⊞ Designing your classe
⊞ Java Data Structures
⊞ Passing the unit tests
⊞ What is wrong with th
⊞ Writing Code Home A
⊞ Written Test Core Jav
⊞ Written Test JEE (1)

How good are your .....?

open all | close all
⊞ Career Making Know-
⊞ Job Hunting & Resum

What if you do the opposite? Rebase the “master branch onto the feature branch”? In other words, you take the changes from the feature-branch into the master.

The rebase moves all of the commits in master onto the tip of feature commits. So, what will happen to all the other developers who are still working with the original master? It will not only invalidate the history on the “master”, but also can break others’ development work. So, before you run rebase, always ask yourself, “Is anyone else looking at this stream of work?”



scm: rebasing “when not to”

Q3. What do you understand by the term patch or pull request?
A3. A patch means change sets you want to communicate and apply to another repository. Nowadays, the GitHub “pull request” makes it really easy to apply patches on GitHub repos, which is useful when you aren’t a direct contributor. A patch is a small file that indicates what was changed in a repository. It’s generally used when someone from outside your team has read-only access but had a good code change available. He/she then creates a patch and sends it to you.

Q4. What is a “sync merge” and when do you perform it?
A4. This is more a SVN terminology. Say you create a new branch from a head to work on some enhancements, and simultaneously some bug fixes were made on the trunk. Suppose that a month has passed since you started working on your new branch and your new feature are not finished yet, but at the same time you know that other people on your team continue to make important bug fixes on the trunk. It’s in your best interest to replicate those changes to your own branch, just to make sure that they integrate well with your

changes. This is done by performing a sync merge, which is a merge operation designed to bring your branch up to date with any changes made to its ancestral parent, which is the trunk in this case.

Q5. What is a fast-forward merge?
A5. This is a Git terminolgy. Git supports two types of merges between branches: fast-forward merge and true merge. When doing a git merge, if only one of the two branches in the merge has changed, a **fast-forward merge** will occur. If both sides have changes, than a true merge will occur. Fast-forward merges create no “merge commit” and the result looks like a rebase.

Q6. What are the pros & cons of merging Vs rebasing in Git?
A6.

Merging Pros:

1) Simple to use and understand.

Merging cons

1) Merges can create a fishbone like history tree, which is a clutter if the need to merge arises simply because multiple people are working on the same branch in parallel.

Rebase Pros:

Simplifies your history, as they are linear in nature.

Rebase Cons:

1) Slightly more complex, especially under conflict situations. As each commit is rebased in order, and a conflict will interrupt the process of rebasing multiple times. For every conflict, you have to resolve the conflict in order to continue the rebase.

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 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 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**

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

Linear & Binary Search in Java ›

Posted in Git & SVN, SDLC

Tags: Free Content

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

 ↑