What are some good strategies to allow deployed applications to be hotfixable? [closed]

Asked 16 years, 2 months ago Modified 2 years, 4 months ago

Viewed 2k times

9

Closed. This question is opinion-based. It is not currently accepting answers.

Want to improve this question? Update the question so it can be answered with facts and citations by editing this post.

Closed last year.

Improve this question

In an ideal world, our development processes would be perfect, resulting in regular releases that were so thoroughly tested that it would never be necessary to "hotfix" a running application.

But, unfortunately, we live in the real world, and sometimes bugs slip past us and don't rear their ugly heads until we're already busy coding away at the next release. And the bug needs to be fixed ***Now***. Not as a

part of the next scheduled release. Not tonight when the traffic dies down. ***Now***.

How do you deal with this need? It really can run counter to good design practices, like refactoring your code into nice, discrete class libraries.

Hand-editing markup and stored procedures on a production server can be a recipe for disaster, but it can also avert disaster.

What are some good strategies for application design and deployment techniques to find a balance between maintenance needs and good coding practices?

deployment **language-agnostic** **release-cycle**

Share

Improve this question

Follow

edited Aug 19, 2022 at 19:35

Joshua

43.1k ● 9 ● 78 ● 148

asked Sep 27, 2008 at 15:27

Eric Z Beard

38.4k ● 27 ● 101 ● 147

3 Answers

Sorted by: Highest score (default)

[Even though we test a lot before we release,] What we do is this:

2

Our SVN looks like this:

```
/repo/trunk/
/repo/tags/1.1
/repo/tags/1.2
/repo/tags/1.3
```

Now whenever we release, we create a tag which we eventually check out in production. Before we do production, we do staging which is [less servers but] pretty much the same as production.

Reasons to create a "tag" include that some of the settings of our app in production code are slightly different (e.g. no errors are emailed, but logged) from "trunk" anyway, so it makes sense to create the tag and commit those changes. And then checkout on the production cluster.

Now whenever we need to *hotfix* an issue, we fix it in `tags/x` first and then we `svn update` from the tag and are good. Sometimes we go through staging, with some issues (e.g. minor/trivial fixes like spelling) we by-pass staging.

The only thing to remember is to apply all patches from `tags/x` to `trunk`.

If you have more than one server, Capistrano (link to capify.org doesn't go to the intended anymore) is extremely helpful to run all those operations.

Share Improve this answer

Follow

edited Aug 19, 2022 at 19:38

Joshua

43.1k ● 9 ● 78 ● 148

answered Sep 27, 2008 at 15:42

Till

22.4k ● 4 ● 60 ● 89

How do you handle testing the fix in development? Usually on a dev box I have a million things pointed at my trunk folder that need to be re-configured. Also, what's the advantage of an SVN tag over simply noting the latest revision number when you release? Thanks! – Eric Z Beard Sep 27, 2008 at 15:48

Good question, I extended my answer and added that we have a staging environment. (RE: difference) IMHO it's the use-case, easier to work with a tag than a revision and as I explained, we make production/staging specific additions before we deply. Hope that helps. – Till Sep 27, 2008 at 15:57

1 @Till The link Capistrano leads to a gambling ad site. I believe this wasn't your intention (back in 2008), was it? – Scheff's Cat Feb 9, 2022 at 17:26

1

One strategy is to heavily use declarative-style external configuration files for the different components.

Examples of this:

- Database access/object-relational mapping via a tool like IBatis/IBatis.NET
- Logging via a tool like JLog/NLog
- Dependency injection via a tool like Spring/Spring.NET

In this way, you can often keep key components separated into discrete parts, hotfix a running application without recompile, and seamlessly use source control (particularly in comparison to stored procedures, which usually require manual effort to source control).

Share Improve this answer

Follow

answered Sep 27, 2008 at 15:51

Ben Hoffstein

103k ● 8 ● 106 ● 121

-1

We divide our code in framework code and business customizations. Business customization classes are loaded using a separate classloader and we have tool to submit changes to a running instance of production. whenever we need a change in any class we change it and submit it to a running instance. the running instance will reject the old classloader and use a new classloader

insance to load the classes again. This is similar to Jboss hot deploy of EJBs.

Share Improve this answer

Follow

answered Sep 27, 2008 at 16:18

Kalpesh Patel

333 ● 3 ● 13