Best practice for test and production environments

Asked 9 years, 2 months ago Active 5 months ago Viewed 28k times

12

1

In the company where I work, we have 2 environments: test and production. We are not currently starting a new environment, because of cost.

Here is the procedure we follow: business makes a feature request, development makes it happen and deploys it on test environment. Then business tests it (UAT), and if it's OK, the feature will be included into next production deployment.

The problem is best practices for test DB. Developers treat test environment as their playground, and sometimes they reset the DB to initial state for testing purposes. On the other hand, business people think that test DB must be stable, and should not be reset. We would like to resolve this issue, and decide if test environment should belong to development team or business team. (Developers don't want business to put their nose in test env., but business team is paying for servers.)

What is the best practice about environments? Can you recommend an article about this?

production-environment test-environments

edited Dec 18 '19 at 19:58
Andrew
21 ● 5

asked Mar 12 '11 at 21:49
SadullahCeran
2,336 ● 4 ● 17 ● 34

4 Answers

Active Oldest Votes

9

At our company there are two databases too, a test and a production database. The test database is mainly used for testing by developers but sometimes for business tests too. This database is refreshed daily using an actual copy of the production database. So this database can be both a playground and a serious testing database. But a third, development, database is the best option. We had one, but it is broken at the moment. But when you get one of those, you should make sure it is refreshed often enough. When developers use it as a playground, it will stray away from the production environment, and its data will be both old and currupt. Because of this, developers won't be able to test well themselves. So make sure you refresh this database periodically (maybe daily too, or at least once a week).

edited Oct 8 '17 at 15:59
Kerem Baydoğan
9,210 ● 33 ● 49

answered Mar 12 '11 at 22:03
GolezTrol
105k ● 10 ● 153 ● 184

6 **Development Database**. `We had one, but it is broken at the moment.` ***story of my life*** – Kerem Baydoğan Oct 8 '17 at 16:03

10

I've worked in many companies, each one with a different set of environments, the one I have like the most had 5 environments:

1) Local: Basically your machine. Here is where you code and test your changes before even asking a peer for review.

2) Dev: If for some reason you cannot test your code locally (Dependencies issues mainly: "My code has neves compiled in my machine but it compiles perfectly in Jenkins/Bamboo/Travis"), then you push your changes to your feature branch in Git and make Bamboo compile it and deploy it to a dev server where you can test (you are still not sure if it will work, so no peer review so far).

3) Staging: You think your code work and you love how it looks like. You create a Pull Request in order for you peers to take a look at it before it gets merged to the master branch. Here they make comments and you fix possible issues, since you are more sure about your changes you make Bamboo deploy it to the Staging environment where more "stable" code lives and more realistic data is stored in whatever database is there. Once deployed another developer/tester can check your changes actually work and make you a "QA Sign off in Staging Env."

4) Stable: Ok, now you are ultimate sure your changes work since you deployed to Staging and nothing got broken. You merge your branch to master and Bamboo compiles master and deploys to another sets of servers in a Stable Environment (no one else should merge to master until you finish your deployment to Production, to avoid a merge of not related merges). This environment should be a replica from Production, data, code and server conditions. Here is where you show your changes to your manager, product owner or the person in charge to validate your work before sending it to production. You get the final approval, everything is good, you are sweaty, you have worked for 30 days in a row to get this change done, your wife divorced you but you are very confident it works perfectly.

5) Production: Where the clients connect to consume the company services, or the final build of your software to send to customers. With a few clicks in Bamboo you make it deploy to Production servers or compile the final build. It is green, everything seems to be ok. You check Splunk looking for errors, everything is good, life is good, you drink another sip of coffee before leaving, you drive home and sleep all weekend with your dog by your side.

It's a happy ending because having so many "Test" environments ensure quality, no change gets to production until EVERYBODY (not just you) are completely sure that changes is working.

answered Jul 19 '17 at 16:53

hmojica
454 ● 5 ● 11

7

If possible, give each developer her own database on her local machine. That way she can do whatever she wants with it without affecting anyone else. This should significantly decrease her desire to play with the test database, providing a more stable environment for business UAT.

3

I believe in order to establish an environment strategy that supports all ALM/SDLC activities 4 requirements should exist:

1) Development environment: to allow Dev to play around with new code/concepts and typically unit test with some basic integration testing using stubs and drivers. This environment should have loose change control procedures and would typically not be anywhere near the same scale as production. This is where the Dev team can build and tear down setups as required.

2) Interop environment: where integration of systems can be further tested and increased capability for non-functional testing I.e might be a resilient env with greater scalability than Dev. I'd see this environment having tighter change control and management. Test would perform integration and system testing in this environment.

3) Reference Architecture: This is what some might call pre-prod but is essentially the same as production in terms of scale and resilience. This would have change control and management procedures akin to prod. This env would support further test activities especially full scale performance testing, failover as well as operational fault triage and maintenance activities once a product is launched to customers.

4) Production: This environment will support live customers and so test activities will be limited once this is the case. This will be fully managed and have strict change management and config management processes.

Hope this helps

edited Sep 16 '16 at 15:38

rado

3,752 ● 3 ● 27 ● 23

answered Apr 12 '13 at 21:41

BenDatlen

31 ● 1