



 Print Article  Close Window

From: www.cio.com

12 Tips to Avoid Gridlock in the Cloud

– David Taber, CIO

April 02, 2012

In Star Wars, Coruscant is an entire planet that's a city. And in the movie, traffic flows in three dimensions as everyone flies around - without any accidents. As it's science fiction, there aren't even any traffic jams.

Things aren't so nice in the real world. For a real thrill, try driving in Bangalore or Rome. I've never seen any traffic accidents in Rome (do the FIATs simply vaporize on impact?), but I've been in taxis there that drove on the wrong side of the street, and on the sidewalk, and skidding around on cobblestones.

In the cloud of science fiction, deployment is smooth and painless, verging on pushbutton. This is true for simple applications that don't have a lot of dependencies or interactions with external systems or third-party applications. And it can even be true for complex cloud deployment if you do the following:

1. Check for driver's licenses. To avoid deployment traffic jams and accidents, administrative certifications really matter (even more than developer certifications in most cases).
2. Have your scrum teams include a process step for architectural review of proposed changes. Some of the most innocent-sounding changes in access controls, validation rules, and even picklist values can have big repercussions. This goes double if you have more than one scrum team working in parallel.
3. If you can find a configuration management (or at least audit) tool that works across most of your cloud technologies, use it. If you can't, use version-managed documents to keep track of what's changing. Google docs work well, particularly if you archive the docs into static files for archiving on a weekly basis.
4. Have unit tests for all modules that have more than 90 percent test coverage and do assertion testing for major positive and negative test cases. For the core business logic, consider using test-driven development.
5. Run all unit tests and record results every day, even if there is no change to your code. Your SaaS platform or third-party software elements may have changed without your knowing about it. If there is any unit test failure, that goes to the scrum team as their highest priority "story."
6. Use ANT or some similar scriptable deployment system on a regular basis, and try to follow the Continuous Integration (CI) best practices of Agile.
7. Have system-level tests that exercise every external interface with representative test data, and that perform a statistical sanity check on the database (e.g., "do we have 50% new customers since the last test run?")

8. Run all system tests and record results at least once a week, even if there is no change to your code. Not only may the platform or third-party software elements have changed, your system's data may have expanded or changed in ways that invoke code paths in previously-untested ways. Test failures go to the scrum team with highest priority.
9. Have several sandboxes (essentially, one per sub-team) which are frequently synchronized / refreshed. The sync/refresh cycle should be an explicitly scheduled item that becomes part of the scrum-team's agenda.
10. Have system administration configuration control practices that prevent experimental changes from persisting beyond the experiment.
11. Have controls that prevent undocumented changes to either sandbox or production systems.
12. Have controls that prevent changes from being applied directly to production systems.

What to Do in the Real World

Science fiction is fine, but cloud vendors are innovating like mad and it's early days for deployment infrastructure and discipline. The best cloud vendors do a good job for their direct customers, but there's not much they can do for other vendors' technology, let alone open source services.

All too many teams simply haven't put in place the deployment infrastructure and disciplines listed above. If you're in that situation, you need to assess the risk of where you are and prioritize corrections accordingly.

If your team is in immediate trouble (e.g., you can't get a bug-fix deployed), the first priority is to get out of your skid. As with a spin, the rule is to focus on getting control of the situation, not actually getting where you want to go. Then, figure out how to simplify the problem by turning things off, reducing the number of variables. Once you have the fire put out, make sure that deployment infrastructure and practices get prioritized ahead of any new feature development. Why? You've built up a technical debt that has to be paid. Make sure that every new project pays a "deployment tax," so that the deployment infrastructure and discipline are not starved for resources and can step up to the challenge as the number of moving parts increases. Otherwise, you'll never get out of your gridlock situation.

David Taber is the author of the new Prentice Hall book, "[Salesforce.com Secrets of Success](#)" and is the CEO of [SalesLogistix](#), a certified Salesforce.com consultancy focused on business process improvement through use of CRM systems. SalesLogistix clients are in North America, Europe, Israel, and India, and David has over 25 years experience in high tech, including 10 years at the VP level or above.

Follow everything from CIO.com on Twitter [@CIOonline](#)

© 2012 CXO Media Inc.