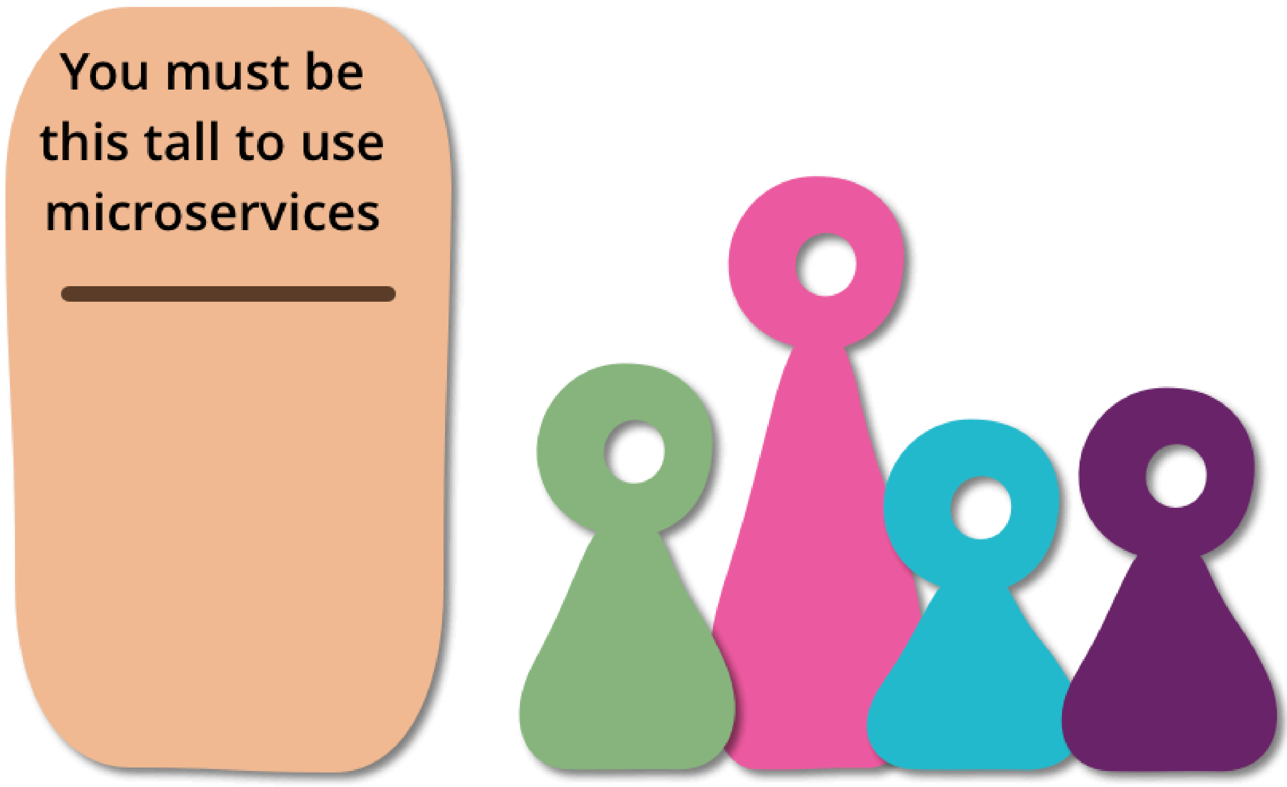
Microservice Prerequisites

28 August 2014

Martin Fowler

MICROSERVICES

As I talk to people about using a microservices architectural style I hear a lot of optimism. Developers enjoy working with smaller units and have expectations of better modularity than with monoliths. But as with any architectural decision there are trade-offs. In particular with microservices there are serious consequences for operations, who now have to handle an ecosystem of small services rather than a single, well-defined monolith. Consequently if you don't have certain baseline competencies, you shouldn't consider using the microservice style.



Rapid provisioning: you should be able to fire up a new server in a matter of hours. Naturally this fits in with CloudComputing, but it's also something that can be done without a full cloud service. To be able to do such rapid provisioning, you'll need a lot of automation - it may not have to be fully automated to start with, but to do serious microservices later it will need to get that way.

Basic Monitoring: with many loosely-coupled services collaborating in production, things are bound to go wrong in ways that are difficult to detect in test environments. As a result it's essential that a monitoring regime is in place to detect serious problems quickly. The baseline here is detecting technical issues (counting errors, service availability, etc) but it's also worth monitoring business issues (such as detecting a drop in orders). If a sudden problem appears then you need to ensure you can quickly rollback, hence…

Rapid application deployment: with many services to manage, you need to be able to quickly deploy them, both to test environments and to production. Usually this will involve a DeploymentPipeline that can execute in no more than a couple of hours. Some manual intervention is alright in the early stages, but you'll be looking to fully automate it soon.

These capabilities imply an important organizational shift - close collaboration between developers and operations: the **DevOpsCulture**. This collaboration is needed to ensure that provisioning and deployment can be done rapidly, it's also important to ensure you can react quickly when your monitoring indicates a problem. In particular any incident management needs to involve the development team and operations, both in fixing the immediate problem and the root-cause analysis to ensure the underlying problems are fixed.

With this kind of setup in place, you're ready for a first system using a handful of microservices. Deploy this system and use it in production, expect to learn a lot about keeping it healthy and ensuring the devops collaboration is working well. Give yourself time to do this, learn from it, and grow more capability before you ramp up your number of services.

If you don't have these capabilities now, you should ensure you develop them so they are ready by the time you put a microservice system into production. Indeed these are capabilities that you really ought to have for monolithic systems too. While they aren't universally present across software organizations, there are very few places where they shouldn't be a high priority.

Going beyond a handful of services requires more. You'll need to trace business transactions through multiple services and automate your provisioning and deployment by fully embracing ContinuousDelivery. There's also the shift to product centered teams that needs to be started. You'll need to organize your development environment so developers can easily swap between multiple repositories, libraries, and languages. Some of my contacts are sensing that there could be a useful MaturityModel here that can help organizations as they take on more microservice implementations - we should see more conversation on that in the next few years.

Acknowledgements

This list originated in discussions with my Thoughtworks colleagues, particularly those who attended the microservice summit earlier this year. I then structured and finalized the list in discussion with **Evan Bottcher, Thiyagu Palanisamy, Sam Newman, and James Lewis**.

And as usual there were valuable comments from our internal mailing list from Chris Ford, Kief Morris, Premanand Chandrasekaran, Rebecca Parsons, Sarah Taraporewalla, and Ian Cartwright.