As I talk to people about using a [microservices architectural style](https://martinfowler.com/articles/microservices.html) 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](https://martinfowler.com/bliki/CloudComputing.html), 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](https://martinfowler.com/bliki/DeploymentPipeline.html) 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](https://martinfowler.com/bliki/DevOpsCulture.html)**. 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.