Figure 1-5. You can use provisioning tools with your cloud provider to create servers, databases, load balancers, and all other parts of your infrastructure.

What Are the Benefits of Infrastructure as Code?

Now that you've seen all the different flavors of IaC, a good question to ask is, why bother? Why learn a bunch of new languages and tools and encumber yourself with yet more code to manage?

The answer is that code is powerful. In exchange for the upfront investment of converting your manual practices to code, you get dramatic improvements in your ability to deliver software. According to the 2016 State of DevOps Report, organizations that use DevOps practices, such as IaC, deploy 200 times more frequently, recover from failures 24 times faster, and have lead times that are *2,555* times lower.

When your infrastructure is defined as code, you are able to use a wide variety of software engineering practices to dramatically improve your software delivery process, including the following:

Self-service

Most teams that deploy code manually have a small number of sysadmins (often, just one) who are the only ones who know all the magic incantations to make the deployment work and are the only ones with access to production. This becomes a major bottleneck as the company grows. If your infrastructure is defined in code, the entire deployment process can be automated, and developers can kick off their own deployments whenever necessary.

Speed and safety

If the deployment process is automated, it will be significantly faster, since a computer can carry out the deployment steps far faster than a person, and safer, given that an automated process will be more consistent, more repeatable, and not prone to manual error.

Documentation

If the state of your infrastructure is locked away in a single sysadmin's head, and that sysadmin goes on vacation or leaves the company or gets hit by a bus,[4] you may suddenly realize you can no longer manage your own infrastructure. On the other hand, if your infrastructure is defined as code, then the state of your infrastructure is in source files that anyone can read. In other words, IaC acts as documentation, allowing everyone in the organization to understand how things work, even if the sysadmin goes on vacation.

Version control

You can store your IaC source files in version control, which means that the entire history of your infrastructure is now captured in the commit log. This becomes a powerful tool for debugging issues, because any time a problem pops up, your first step will be to check the commit log and find out what changed in your infrastructure, and your second step might be to resolve the problem by simply reverting back to a previous, known-good version of your IaC code.

Validation

If the state of your infrastructure is defined in code, for every single change, you can perform a code review, run a suite of automated tests, and pass the code through static analysis tools—all practices that are known to significantly reduce the chance of defects.

Reuse

You can package your infrastructure into reusable modules so that instead of doing every deployment for every product in every environment from scratch, you can build on top of known, documented, battle-tested pieces.[5]

Happiness