questions, especially about how to provision the servers and install the agent software on them in the first place.

Ansible, CloudFormation, Heat, Terraform, and Pulumi are all masterless by default. Or, to be more accurate, some of them rely on a master server, but it's already part of the infrastructure you're using and not an extra piece that you need to manage. For example, Terraform communicates with cloud providers using the cloud provider's APIs, so in some sense, the API servers are master servers, except that they don't require any extra infrastructure or any extra authentication mechanisms (i.e., just use your API keys). Ansible works by connecting directly to each server over SSH, so again, you don't need to run any extra infrastructure or manage extra authentication mechanisms (i.e., just use your SSH keys).

Agent Versus Agentless

Chef and Puppet require you to install *agent software* (e.g., Chef Client, Puppet Agent) on each server that you want to configure. The agent typically runs in the background on each server and is responsible for installing the latest configuration management updates.

This has a few drawbacks:

Bootstrapping

> How do you provision your servers and install the agent software on them in the first place? Some configuration management tools kick the can down the road, assuming that some external process will take care of this for them (e.g., you first use Terraform to deploy a bunch of servers with an AMI that has the agent already installed); other configuration management tools have a special bootstrapping process in which you run one-off commands to provision the servers using the cloud provider APIs and install the agent software on those servers over SSH.

Maintenance

You need to update the agent software on a periodic basis, being careful to keep it synchronized with the master server if there is one. You also need to monitor the agent software and restart it if it crashes.

Security

If the agent software pulls down configuration from a master server (or some other server if you're not using a master), you need to open outbound ports on every server. If the master server pushes configuration to the agent, you need to open inbound ports on every server. In either case, you must figure out how to authenticate the agent to the server to which it's communicating. All of this increases your surface area to attackers.

Once again, Chef and Puppet do have varying levels of support for agentless modes, but these feel like they were tacked on as an afterthought and don't support the full feature set of the configuration management tool. That's why in the wild, the default or idiomatic configuration for Chef and Puppet almost always includes an agent and usually a master, too, as shown in Figure 1-7.

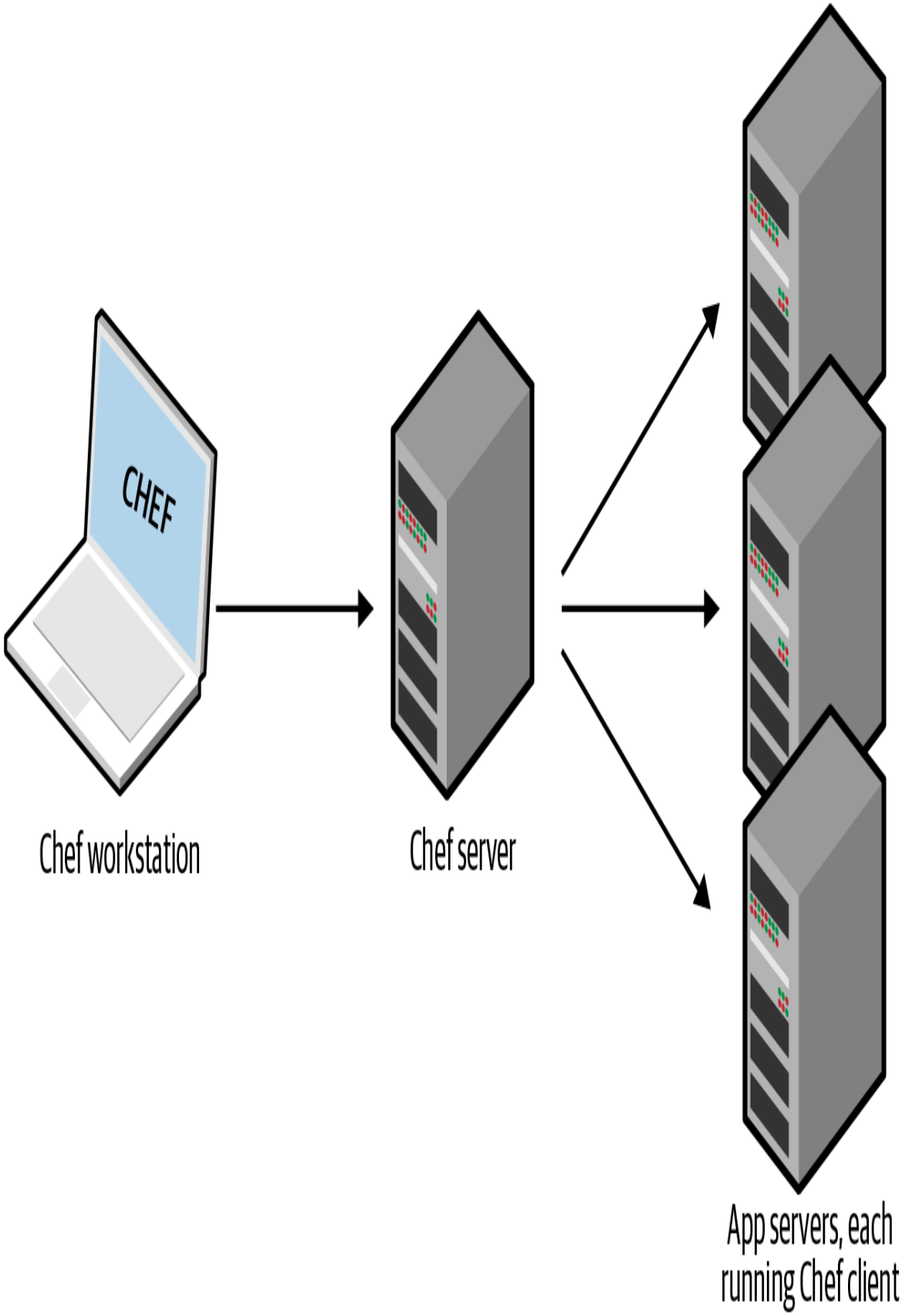
CHEF
Chef workstation
Chef server
App servers, each
running Chef client

Figure 1-7. The typical architecture for Chef and Puppet involves many moving parts. For example, the default setup for Chef is to run the Chef client on your computer, which talks to a Chef master server, which deploys changes by communicating with Chef clients running on all your other servers.

All of these extra moving parts introduce a large number of new failure modes into your infrastructure. Each time you get a bug report at 3 a.m., you'll need to figure out whether it's a bug in your application code, or your IaC code, or the configuration management client, or the master server(s), or the way the client communicates with the master server(s), or the way other servers communicate with the master server(s), or…

Ansible, CloudFormation, Heat, Terraform, and Pulumi do not require you to install any extra agents. Or, to be more accurate, some of them require agents, but these are typically already installed as part of the infrastructure you're using. For example, AWS, Azure, Google Cloud, and all of the other cloud providers take care of installing, managing, and authenticating agent software on each of their physical servers. As a user of Terraform, you don't need to worry about any of that: you just issue commands, and the cloud provider's agents execute them for you on all of your servers, as shown in Figure 1-8. With Ansible, your servers need to run the SSH daemon, which is common to run on most servers anyway.

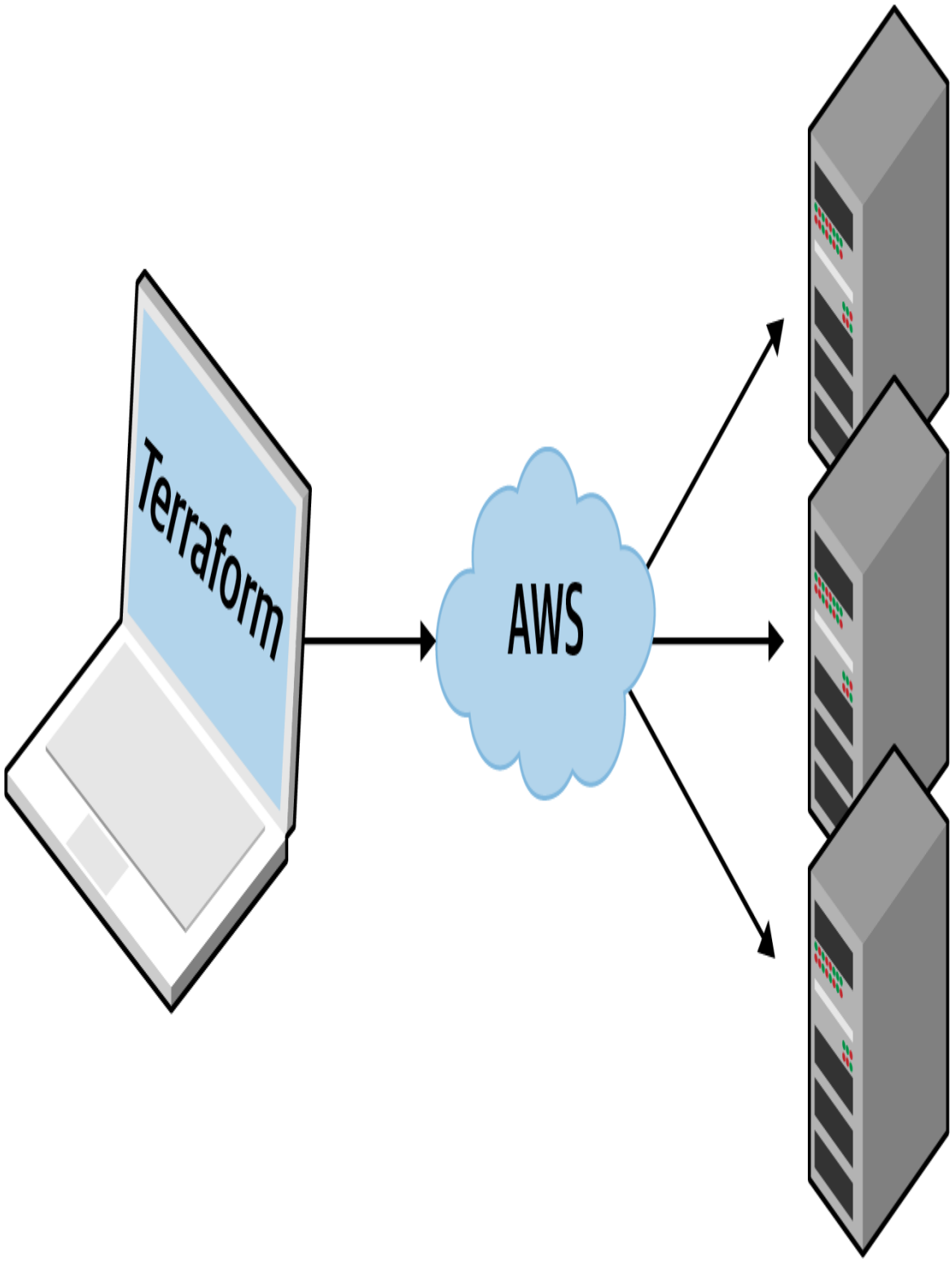

Figure 1-8. Terraform uses a masterless, agentless architecture. All you need to run is the Terraform client, and it takes care of the rest by using the APIs of cloud providers, such as AWS.