Why not install SSH on every server

Mark Bools

December 7, 2020

Last Modified: February 25, 2021

Abstract

Tempting though it may be, blindly enabling SSH on all your servers is a huge mistake.

SSH is a convenient secure method of accessing your servers, so why not enable it on all your servers so that you can log in and check the server our when you have a problem?

To be clear, I'm talking here primarily of installing SSH to allow interactive login. SSH used as a secure means non-interactive commands (typically mediated by a configuration management tool like Ansible) is of less concern.

For all it's good points SSH is a vulnerability. Firstly, it gives the user a means to interact directly with the server. If that user has any privileges on that server this means you have introduced the potential for human error in managing that machines configuration. It is too easy to accidentally (or maliciously) make modifications to a machines configuration and then overlook the all important step of documenting that change so that subsequent development knows that the configuration has changed.

A simple example. You have a communication issue between to machines in your production environment, this is causing problems for some users so your administrator logs onto the server they believe to be the problem and investigate. Their investigation soon reveals a small mistake in the server's host firewall. In order to mitigate the production issue the administrator 'tweaks' the firewall. All is now well with the production system and the issue is resolved.

Many months later an new development effort is launched and as part of that effort a new environment is being built to reflect the current production environment. The team use the current configuration documentation to build the new system. After a few days the new system is brought online and the development proceeds but a few days into development the team experiences some 'odd' behaviour not currently seen on the production system. Investigation uncovers some issues with the inter-machine communication and ultimately with one server's host firewall. Deeper investigation uncovers a discrepancy between the production server and the development server firewalls.

All this time and energy is wasted. Furthermore confidence in the accuracy of the current configuration is eroded. What other discrepancies between the development and production environments remain undiscovered and what impact will they have on the project?

In a real life environment this simple situation is often repeated multiple times and the cumulative costs of discrepancies mount quickly. In many cases the plan to duplicate the production configuration are abandoned as a lost cause and the new development is simply planned as a new suite of servers largely separate to the existing infrastructure.

All this stemmed from a well intentioned change made by an administrator with direct access to the server via SSH.

The problem, of course, is not SSH itself but the provision of direct access to the server. Since the majority of Linux installations provide direct access via SSH I think it reasonable for the purposes of this article to use the one as a proxy for the other.

So, if one is denying direct access to the servers how are engineers to diagnose faults?

The first thing we need is access to information about the server. There are three things we might need when diagnosing a problem:

1. The machine's current configuration.
2. The machine's logs.
3. The machine's operational state.

The first of these is best preserved if we follow good infrastructure as code and continuous deployment practice (the polar opposite to having direct SSH access). Strict adherence to these practices means engineers can, at any time, look up, and in most instances replicate, the current deployed configuration of the server.

The second can be provided by consolidated real-time logging within our infrastructure. Engineers can refer to the centralised logging facility to see both current and historic logs generated from a particular machine.

Finally, in a similar fashion to the logging, we have centralised real-time analytics for our system. Engineers can reference current and historical time series analytics (things such a process load, running threads, memory use, etc.) These can be correlated with logs to assist in diagnosis.

All of this information can be made available to engineers with no need for direct access to the machine itself. Furthermore, if configuration information is provided through infrastructure as code and continuous delivery we should be able to fairly efficiently build a duplicate server to which engineers can have direct access.

Does this mean we *never* need direct access to a server? That rather depends, but I contend that the number of times direct access is required, assuming the above conditions are met, is vanishingly small.