VOLUME 1

THE PRACTICE OF SYSTEM AND NETWORK ADMINISTRATION

THIRD EDITION

THOMAS A. LIMONCELLI • CHRISTINA J. HOGAN • STRATA R. CHALUP

The Practice of System and Network Administration

Volume 1

Third Edition

the IP address space of a particular VLAN must change, usually due to mergers, reorganizations, or the need to physically move machines. In each case the project is made unnecessarily complex due to hardcoded IP addresses. If a service doesn't work, debugging becomes a hunt for the mystery location of a hardcoded IP address buried deep in a configuration file.

There are some exceptions to this rule. First, DNS clients must be configured using IP addresses, due to the Catch-22 that would happen if DNS clients needed to use DNS to find the IP address of a DNS server. NICs and network equipment are another exception for similar reasons. However, there are still a couple of ways that you mitigate against an eventual need to change IP addresses. Look for mechanisms that allow you to use a different service-based IP address in the configuration files, rather than the primary IP address of the machine itself.

Using IP Aliases

There are some other cases where IP addresses are commonly hardcoded into configurations. For example, IP addresses are often used in firewall rules. Likewise, the IP addresses of DHCP servers are often found in the router configurations. Routers are configured to relay broadcast DHCP requests from a particular network to a specified set of DHCP servers. This is called a DHCP relay or IP helper configuration.

DHCP servers can usually be configured with a second IP address (often known as a secondary IP address or an IP alias) that is used for the service, and can be easily moved to another machine. A similar approach can often be used for other services that require firewall rules, with the secondary IP address being used for the service and in the firewall rules, making it much easier to move the service to another machine, along with the IP alias. When this secondary IP address is a loopback address that is advertised onto the network as host route, then it is truly independent of the VLAN, and can be easily moved, even to a completely different datacenter.

17.6 Support

During the planning and engineering phase, consider how the service will be supported during its lifetime. Support includes technical aspects such as detecting problems, patching, upgrading, performing backups, and scaling the service. It also includes people and processes, such as identifying and training the support groups, developing mechanisms for customers to report problems and submit requests (and how those problems and requests should be handled), and providing documentation for the customers and the support teams.

You need to know when there are problems with the service and when you should be thinking about scaling it up—therefore the service must be monitored. You will need to put some thought and effort into engineering the appropriate monitoring solution. You should also consider which requests users may make in relation to this service, and how those will be handled. Which automation and processes should be in place? Which documentation is needed to support the service?

17.6.1 Monitoring

It isn't a service if it isn't monitored. If there is no monitoring, then you're just running software. A service should be monitored for availability, problems, performance, and capacity-planning. Monitoring is covered in more detail in Chapter 38, "Service Monitoring."

The helpdesk, or front-line support group, must be automatically alerted to problems with the service so that they can start fixing them before too many people are affected. A customer who notices a major problem with a service and has to call in to report it before the service provider discovers and begins to fix the problem is getting a very low standard of service. Customers do not like to feel that they are the only ones paying attention to problems in the system. Conversely, problems you can detect and fix before they are noticed are like trees that fall in the forest with no one around to hear them. For example, if an outage happens over the weekend and you are alerted in time to fix it before Monday morning, your customers don't even need to know that anything went wrong. (In this case, you should announce by email that the problem has been resolved, so that you receive credit. See Section 49.2.)

Likewise, the SA group should monitor the service on an ongoing basis from a capacity-planning standpoint. Depending on the service, capacity planning can include network bandwidth, server performance, transaction rates, licenses, and physical-device availability. As part of any service, SAs can reasonably be expected to anticipate and plan for growth. To do so effectively, usage monitoring needs to be built in as a part of the service.

During the planning and engineering phase of the project, you need to define what should be monitored, and how. You need to define which events should trigger alerts, and what the priority of those alerts should be. You need to work with the monitoring team to understand how this service gets integrated into the existing tools, and which work is required on both sides to make that happen.

You also need to define the process for adding new components into the monitoring system as the system is scaled up or upgraded. Likewise, you need to define how components are removed from the monitoring system as they are retired or repurposed. Ideally these changes should be automated, and driven

from the inventory system. However, that means that this automation needs to be developed, and the processes need to be clearly defined and documented.

17.6.2 Support Model

The support model needs to specify who supports the various components of the service, and what the OLAs and SLAs for each component are. The OLA and SLA for the service need to take into account the OLAs and SLAs of the constituent parts, as well as the design of the service itself.

For example, if a service has components in multiple datacenters around the world, who provides hands-on support in the datacenters, and what are their SLAs? What are the vendors' SLAs for the various hardware components—do these differ in different locations? What is the SLA for WAN connectivity? Who provides local network support, and who provides WAN support? Who supports the OS that the application runs on? Who supports the application itself? What are the processes, contact information, SLAs, and escalation paths for all of these groups?

In a small or midsize company, many of these roles will be performed by the same people, who all know each other. In large companies, there will be different teams supporting each component, they won't all know each other, and the processes and communication paths may not be clear. Make sure that all of this information is tracked down and documented in advance, rather than finding out in the midst of an incident that you do not know how to contact the on-site support staff in a remote location.

17.6.3 Service Request Model

For most services, the local engineering plan should also include a service request model (SRM). The SRM defines which requests users can make relating to this service, who can make those requests, how they make those requests, which approvals are required, who acts on the requests, which change control policies apply, and which SLA is used for turning around those requests.

Because each service is different, work with the vendor and stakeholders to define which tasks are required, document them, and then practice them in a test environment. Try to estimate the volume of requests of each type, and understand from the stakeholders what would trigger a request of each type and what kind of turnaround time they would expect on each request.

Consider whether it makes sense to automate these requests. For example, if a user-add request will always be generated as a result of a new hire, can you link to the HR system, so that these requests are automated? This will eliminate the need for someone to remember to submit that request, and someone else to process it. Likewise, user-deletes should occur automatically when someone leaves

the company, and perhaps when someone moves to a different role in a group that does not need access. Alternatively, moves could generate access review tasks for the new manager and the service owner.

17.6.4 Documentation

As part of support planning, operational procedures must be defined. These are different for every service, but generally include backups and restores, business continuity or disaster recovery plans, tasks related to onboarding new users, disconnecting users who are leaving, performing periodic tasks, and anything else required to keep the service running.

For each routine task, there must be a runbook that details how the task should be performed. Define the runbooks by performing each task in a test environment. What is learned through this process may lead to changes in the architecture or local engineering plan. For example, you might discover that backup and restore procedures were not considered until after the architecture was defined. This may require changes to the architecture. Alternatively, what is learned may be a minor point, but one that requires changes in the local engineering plan—for example, to define that each machine must have an additional fiber interface on a dedicated backups network. This is an iterative process.

The documentation must also include the steps to take when debugging the service. This documentation should cover which data needs to be gathered, and how, as well as steps to examine that data and use it to debug the problem. There should be documentation for the helpdesk telling these personnel how to perform basic health checks on the service, verify problems that might be reported by end users or the monitoring system, and debug and resolve the most common problems. Chapter 29, "Debugging," describes the steps involved in debugging problems. The documentation should describe how to follow those steps for this service.

The helpdesk documentation should also specify which information needs to be gathered and passed on to the next support level if the issue remains unresolved and needs to be escalated. The documentation for the more senior SA team must include the vendor's contact information and support contract number, all the components that the service depends on, and contact information for all the other teams that support those components. The contact information for other teams should be provided through a link to a well-known team support page that each team maintains itself, rather than duplicating information that may get stale into many different support documents.

17.7 Summary

Service engineering involves designing how a product will be configured and used in your environment. With a simple service, it is a matter of installing the software. Of course, most services are not simple. In such a case, we must design the service architecture and local engineering plans.

The service architecture is the high-level design. It describes how the parts will work together, which resources are required, and so on. Local engineering plans describe specifics for a particular device, such as its model, name, and IP address.

While a lot of science and methodology goes into such engineering, a few rules of thumb are helpful: Follow vendor recommendations, mirror your boot disk, use service-class hardware, don't run a service on a machine under your desk, restrict console access, and leverage the existing infrastructure as much as possible.

Service architecture should strive for simplicity. Revise the design over and over, tweaking and improving it. Consider how the dependencies will affect the service. Consider alternative plans that reduce dependencies by examining failure domains. We can align dependencies to make a system less susceptible to failures. We can align along hardware failure domains, service failure domains, and location failure domains.

When possible, do not expose actual hostnames to users. Instead, expose hostname aliases that can be moved to other machines. Do not hardcode IP addresses unless doing so is unavoidable.

The service architecture should be supportable. It should indicate how the system will be monitored, how it will be accessed for administrative purposes, and so on. Documentation should include a runbook for common tasks, expected failure modes, and debugging typical problems.

Exercises

1. What is a service architecture? What is a local engineering plan? How are they related and/or different?
2. Why is simplicity important in service engineering?
3. What are primary and external dependencies?
4. Which support services improve our ability to operate a service?
5. Pick a service you are familiar with, and list its primary and external dependencies.
6. What is dependency alignment? How can it be used to make a system less susceptible to failures?

7. For a service you are familiar with, describe how its dependency alignment could be improved.
8. Of the rules of thumb listed in Section 17.1, which are the most important? Which were the most surprising to you?
9. Pick a commercial or open source product you are familiar with, and design a service architecture for deploying it to your organization.
10. Suppose you were part of the team described in the "Branch Office Prototypes" case study in Section 17.2. How would you apply the small batches principle of Chapter 2, "The Small Batches Principle," to that situation? Remember that the small offices are scattered around the world.