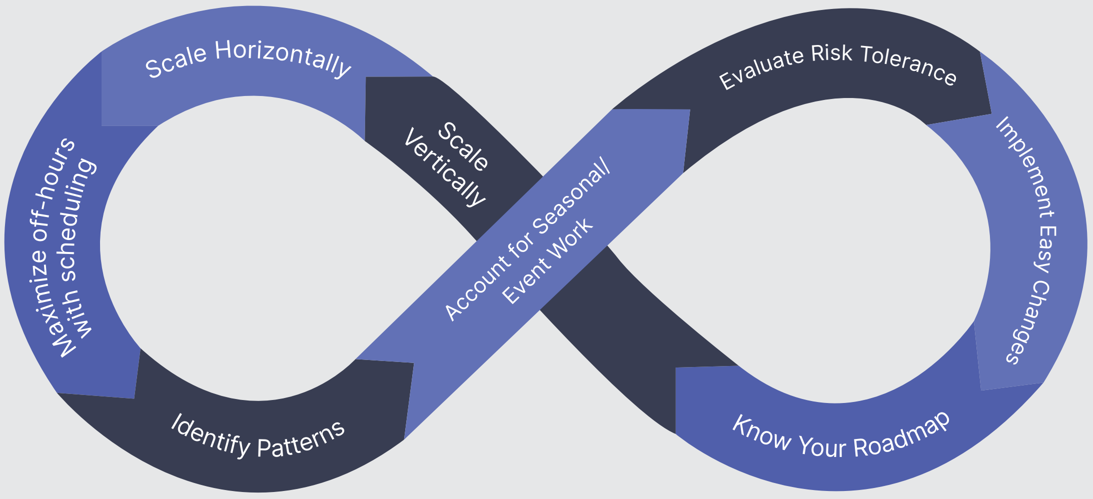
MacStadium

WHITE PAPER

DevOps Capacity Planning

How to determine what hardware you need to drive maximum DevOps efficiency and faster deployments.



Author: Sara Bobo, Developer Productivity and Infrastructure Expert

Table of Contents

Introduction

Resource demand in systems is rarely static across time; as a system's usage grows through increased volume, new features, or more intensive requests, eventually the existing compute resources will no longer be sufficient to support that application or service. The consequences of failing to keep up with growing demand may include performance degradations or even full outage of the service, which can be costly to resolve and harmful to the business or organization.

This white paper serves as an overview of capacity planning for organizations that manage hardware resources for the purposes of running applications or services, both external user-facing and in the developer productivity space.

We'll take a look at assessing and optimizing current utilization, gathering information needed to confidently forecast future needs, and calculating the amount of additional resources that will be required to meet those needs.

Many of the concepts and best practices discussed in this white paper can be applied to hardware capacity planning in a general sense, but here we are focusing specifically on infrastructure for developer productivity tooling, such as Continuous Integration (CI) systems.

“Failure to keep up with a growing DevOps demand may result in performance degradations, long queues, or even full outage of the service, which all result in slower product and feature releases to customers.”

What is capacity planning?

Capacity planning is the process of determining the amount of resources needed to keep a system healthy looking forward. In the context of software, this frequently means determining the appropriate number and configuration of servers or other compute resources to handle an increase in usage. Because of the ever-changing demand on these resources, capacity planning should be done on a regular cadence, alongside other product and operational planning processes.

In a world with infinite budgets, capacity planning would be easy: throw as many high-powered machines at the problem as possible. In reality, budgets are finite, so it is important to carefully determine how many machines, and of what hardware configuration, to balance addressing the business risks resulting from under-provisioning with addressing the unnecessary expense resulting from grossly over-provisioning.

Capacity planning is of enormous importance when an organization is working with on-premise hardware, but it is necessary in cloud environments as well. In platforms such as AWS or GCP, it is possible to scale a system up and down on demand, but eventually even in these environments engineering teams may find themselves boxed in without adequate forethought. Additionally, Apple's EULA mandates that hosted macOS machines and Mac compute may be leased for no less than 24 hours at a time, which prevents vendors from providing the minute-by-minute, on-demand scaling offered in other cloud compute environments. Without full flexibility to conjure up the exact resources required at any given moment, or the ability to scale down once they are no longer needed, teams need to be able to anticipate future needs and estimate what resources will be required.

“Capacity planning is of enormous importance when an organization is working with on-premise hardware, but it is necessary in cloud environments as well.”

In each of the following sections, we'll be returning to an example scenario to explore how the concepts and best practices more concretely:

Alex leads a team that owns a CI system that includes a Jenkins instance and a cloud kubernetes cluster used for deploying worker agents that handle test and build workloads, on-demand. The system is beginning to show signs of strain every afternoon, and even more so right before the weekly release. Developers are already complaining about the long wait times before the weekly release, which are forcing them to submit changes earlier and earlier to avoid getting stuck in the queue. Alex realizes that the current system likely won't be sufficient to handle demand in the coming year, and without scaling there are likely to be long CI queue times or even complete outages for internal customer teams. They want to get their director onboard with a procurement request for approval of additional capacity, but they want to be sure they're not asking for too many (or too few) machines. They also know they should provide justification for their request in the form of metrics or other data.

ALEX

Senior DevOps Engineer

JavaScript Java Swift Python

Understand your current system utilization

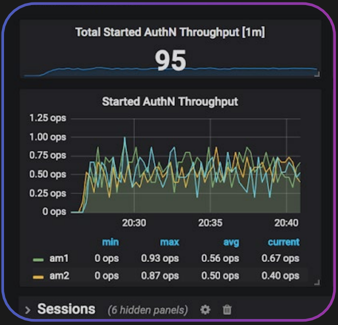
Before considering future needs, it's crucial to take stock of the current state of the system. Talking directly with users and determining their current pain points can be enormously helpful in locating existing performance bottlenecks. It is just as important to keep tabs on user experience for internal customer teams (such as other developers within the organization) as for external end users. Determining and monitoring key metrics can help the owning team assess the impact of their efforts.

✓ Identify usage and performance patterns

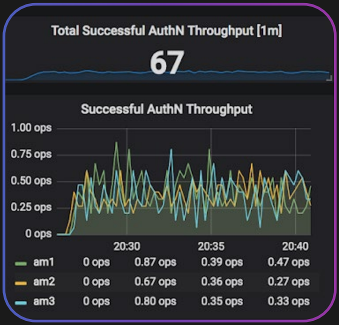
Organizations should be monitoring usage metrics at the individual service or app level, such as volume of requests, or workload duration, to get an idea of usage demands and patterns. For example, an organization with developers concentrated within one or several adjacent time zones will likely see a peak in CI traffic during the workday, with little traffic after work hours and on the weekends. Understanding how the app is currently being used and at what volume is necessary for determining what resources may be needed in the future as those metrics change and grow over time.

KEY POINT

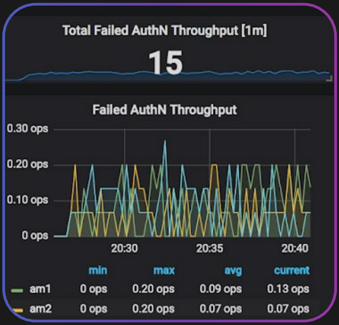
Organizations should be monitoring usage metrics at the individual service or app level, such as volume of requests, or workload duration, to get an idea of usage demands and patterns.



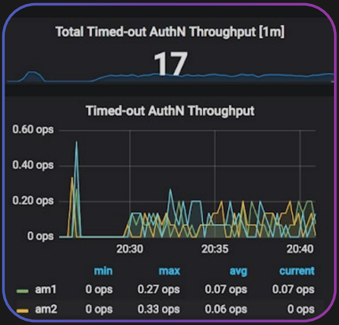
	min	max	avg	current
am1	0 ops	0.93 ops	0.56 ops	0.67 ops
am2	0 ops	0.87 ops	0.50 ops	0.40 ops


Total Successful AuthN Throughput [1m]
67
Successful AuthN Throughput
1.00 ops
0.75 ops
0.50 ops
0.25 ops
0 ops
20:30
20:35
20:40

am1	0 ops	0.87 ops	0.39 ops	0.47 ops
am2	0 ops	0.67 ops	0.36 ops	0.27 ops
am3	0 ops	0.80 ops	0.35 ops	0.33 ops



	min	max	avg	current
am1	0 ops	0.20 ops	0.09 ops	0.13 ops
am2	0 ops	0.20 ops	0.07 ops	0.07 ops


Total Timed-out AuthN Throughput [1m]
17
Timed-out AuthN Throughput
0.60 ops
0.40 ops
0.20 ops
0 ops
20:30
20:35
20:40

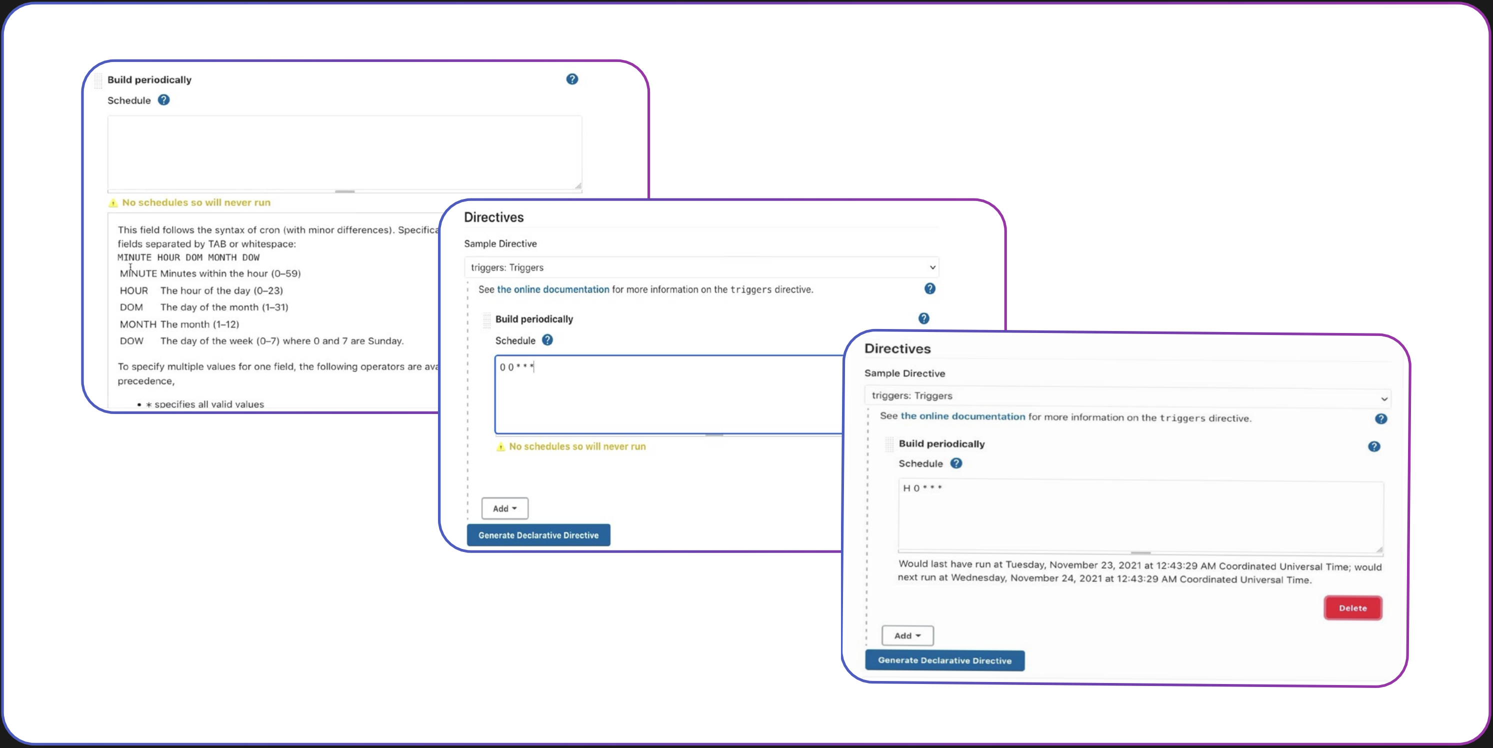
	min	max	avg	current
am1	0 ops	0.27 ops	0.07 ops	0.07 ops
am2	0 ops	0.33 ops	0.06 ops	0 ops

Maximize off-hours with scheduling

Scheduled and automated workloads should be audited to determine that all utilization is actually beneficial to the organization; it is very common for teams to forget about old jobs that continue to use resources on a daily basis. Additionally, identifying scheduled jobs that are not time-sensitive should be done to ensure that their schedules have them run during hours when the system typically sees lower utilization. Paring down on unnecessary utilization, generally and during peak usage times, makes it possible to identify actual resource needs and free up existing resources.

KEY POINT

- Audit all scheduled jobs
- Eliminate old, unnecessary jobs
- Reschedule non-time-sensitive jobs to low-usage hours



Build periodically

Schedule

No schedules so will never run

This field follows the syntax of cron (with minor differences). Specifica
fields separated by TAB or whitespace:

```
MINUTE HOUR DOM MONTH DOW
```

MINUTE	Minutes within the hour (0–59)
HOUR	The hour of the day (0–23)
DOM	The day of the month (1–31)
MONTH	The month (1–12)
DOW	The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are ava
precedence,

- ∗ specifies all valid values

Directives

Sample Directive

triggers: Triggers

See the online documentation for more information on the `triggers` directive.

Build periodically

Schedule

```
0 0 * * *
```

No schedules so will never run

Directives

Sample Directive

triggers: Triggers

See the online documentation for more information on the `triggers` directive.

Build periodically

Schedule

```
H 0 * * *
```

Would last have run at Tuesday, November 23, 2021 at 12:43:29 AM Coordinated Universal Time; would next run at Wednesday, November 24, 2021 at 12:43:29 AM Coordinated Universal Time.

Implement easy process changes

To further optimize the use of current resources, consider what other lower-effort or lower-cost steps may be available to reduce capacity usage through eliminating unnecessary volume, or by improving performance of workloads, which reduces the duration of time a given machine is in use to complete a task. Engineers close to the system likely have a wish list of things that they would fix if they had the time to do so; soliciting these ideas from the team may present opportunities to make more efficient use of current resources.

Back to our example:
Alex has spent time analyzing current usage and auditing the jobs handled by their CI system, removing unneeded workloads and scheduling others off-peak. They are now able to show with the metrics they've gathered that even with these reductions, the size of their worker cluster is not sufficient for handling all developer jobs during times of peak traffic – which correspond with the slowdowns they noticed in the afternoons and before release.

KEY POINT

Engineers close to the system can help identify unnecessary volume and areas for improved workload performance.

Predict your future capacity needs

Know your roadmaps

After gaining a clear picture of actual current resource utilization requirements, we can begin to think about future needs. Building a capacity plan is necessarily a collaborative process and should include consideration for the bandwidth of the teams directly involved in any work on the system, as well as the plans of the system's users and any upstream dependencies. A good place to start is becoming more familiar with the organization's roadmap and goals. For a product team, this might look like staying on top of upcoming feature launches. For teams with internal customers, find out their projected headcount, project plans and the anticipated demand on your systems.

KEY POINT

- For a product team – stay on top of upcoming feature launches.
- For teams with internal customers – monitor the projected headcount and project plan timelines.

Account for seasonal or event-driven activity

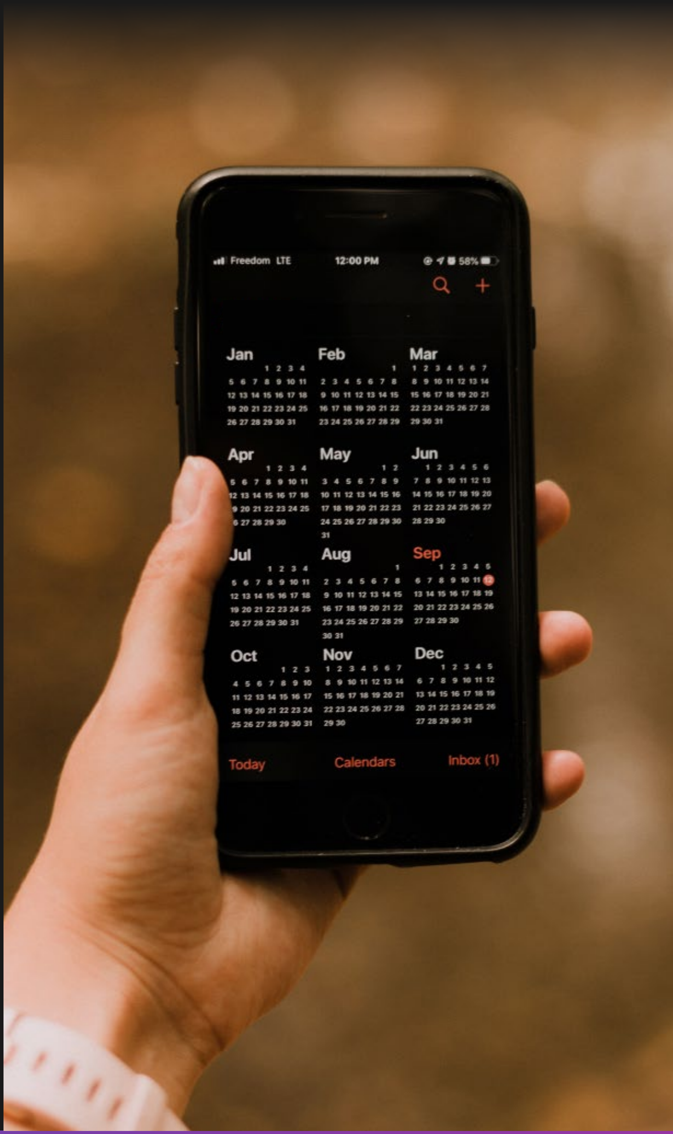
It can also be helpful to think about events outside of the organization itself and how they may impact utilization. For example, a news website might see an increase in traffic during an annual global event, or an ecommerce company may experience an annual peak during the holiday shopping season. These events may bring about periods of high developer activity as teams race to get features done in time. Looking at historical data from previous such events can be enormously helpful in predicting utilization for future cycles.

When talking with their customer organization, Alex learns that the developer teams that use the CI system are projected to grow by about 20% over the next year. Additionally, one of the teams mentions that they're planning on introducing several new crucial test jobs that will run any time an engineer updates their code review, increasing the number of workers in use for each commit by 10%.

Using the information Alex has gathered about current resource utilization, they are able to put together a prediction of the number of additional worker machines they'll need, and they are able to point to the data they used to reach that number. In this case, they are expecting to see demand increase by about 32% at peak over the coming year – all customers will be using 10% more workers to run tests on each commit, multiplied by the increase in developers.

KEY POINT

Look outside your organization to identify the economic trends, seasonal holidays, or other global events that could impact your team's needs.



Plan your targeted resource growth

Knowing the forecast for future utilization will help paint a general sense of how resources might need to scale over that same period, but by itself it doesn't provide concrete estimates on how many more machines may be required.

Evaluate your organization's risk tolerance for each system

Ultimately, the system in question exists to serve a business function, and its reliability is key to performing that function well. Depending on exactly what that system is, periods of service degradation or complete outage present varying levels of business risk. Tolerance for that specific risk (as well as budget) should be used to guide the target percentage of utilization at peak.

For example, if the system in question is necessary for the operation of the core application, or in the critical path of deploying changes, there may be low tolerance for hitting 100% capacity. In planning for that system, again using historical data about the size of traffic spikes, the organization could decide on a lower target utilization percentage, for example 80% utilization at peak, with the remaining 20% serving as buffer in the event of unusual spikes.

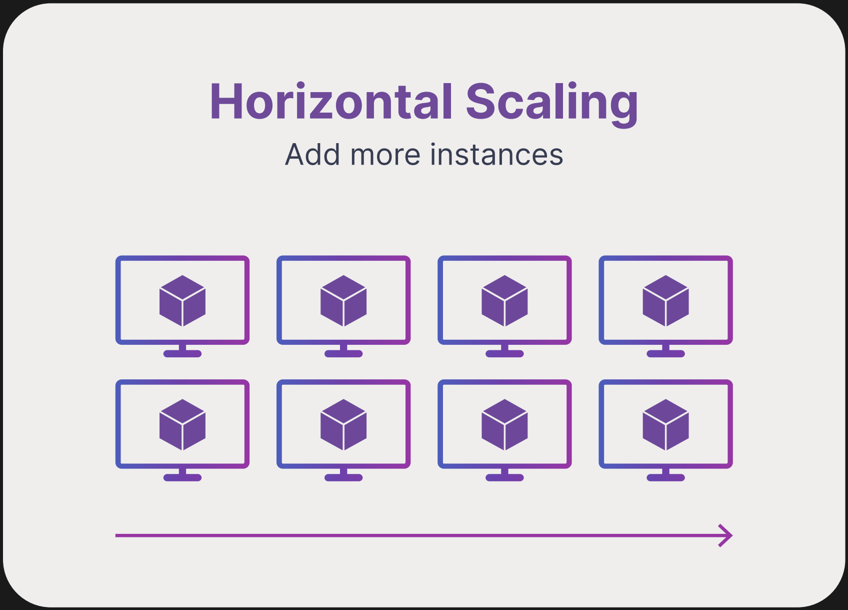
KEY POINT

Not all systems can be top priority. Identify the business impact of each system and plan your capacity accordingly.

Calculate the benefits of horizontal scaling

Calculating the number of machines needed to scale horizontally is relatively straightforward. If we are regularly hitting capacity but need to target a lower percent utilization, we'll need to divide the current number of machines by the target, then multiply by the forecasted increase in demand.

For example, a pool of 50 servers is regularly hitting 100% capacity utilization, but the target utilization is 90%. To reach that target, the pool would need to scale to 50 / .9, or 56 servers, rounded up. Then, if we expect demand to increase by 25 percent in the coming year, the pool would need to further scale to 56 * 1.25, or 70 servers to reliably support business needs.



KEY POINT

In platforms such as AWS or GCP, it is possible to scale a system up and down on demand, but eventually even in these environments engineering teams may find themselves boxed in without adequate forethought. Additionally, Apple's EULA mandates that hosted macOS machines and Mac compute may be leased for no less than 24 hours at a time, which prevents vendors from providing the minute-by-minute, on-demand scaling offered in other cloud compute environments.

Thinking about scaling vertically is a bit more involved. Profiling workloads and looking at metrics for individual machines, such as CPU, memory, disk IO, and network usage can provide insight into resource bottlenecks and indicate whether it might be appropriate to consider migrating to more powerful hardware. If it is possible to gain access to one or more machines of the configurations available to you, running sample workloads many times across the different hardware profiles will provide you with benchmarks to help you evaluate what performance improvements you might expect to see.

Faster workload run-times can translate into lower capacity utilization in a couple of different ways. On more powerful hardware, it may be possible to increase the number of workloads that may be handled simultaneously by a single machine. Additionally, reducing the amount of time each machine is in use for each workload increases the overall throughput of the system – more workloads may be run in a given period of time.

When calculating the anticipated benefit from vertical scaling, it's important to also account for any overhead – while tests may execute faster on the worker agent, any surrounding infrastructure or external dependencies may not see the same benefit. For example, consider a workload with an initial average duration of 10 minutes, where only 8 of those minutes is spent executing tests, while the other two are spent on setup and teardown overhead. While the tests themselves may execute say 30% faster, this will only translate to 24% improvement in overall runtime as the time spent on overhead remains the same.

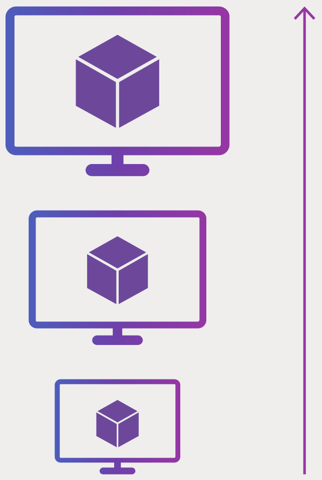
In our example, Alex has discovered that upgrading worker machines to a more powerful machine shows a 33% reduction in end-to-end job duration. Even though they are expecting demand to increase by 32%, they may be better served by upgrading their cluster rather than scaling horizontally – their customer teams will certainly appreciate the shorter wait.

KEY POINT

Don't forget about task overhead! More powerful machines may make quicker work of the task at hand, but often won't improve the time spent on task overhead (e.g., setup and teardown of the task).

Vertical Scaling

Increase size of instance [RAM, CPU, etc.]



Summary

Capacity planning is the process of determining the appropriate number and configuration of servers or other compute resources to handle expected fluctuations in usage.

✓ Understand your current system utilization.

- Identify usage and performance patterns.
- Maximize off-hours with scheduling.
- Implement easy process changes.

✓ Predict your future capacity needs.

- Know your roadmaps.
- Account for seasonal or event-driven activity.

✓ Plan your targeted resource growth.

- Evaluate your organization's risk-tolerance for each system.
- Calculate the benefits of horizontal scaling.
- Calculate the benefits of vertical scaling.

Effectively planning and optimizing your DevOps process compute resources is critical to development team satisfaction and overall business performance. And like the DevOps process itself, DevOps capacity planning an ongoing process. Capacity planning should become a routine part of your organization's overall quarterly and annual planning process. Regularly assessing current and future utilization, organizations can scale their available resources to keep up with upcoming demand and guard against business risk.