CHAPTER 5

# How to Lead a Team

# 第五章 如何領導團隊

**Written by Brian Fitzpatrick**

**Edited by Riona MacNamara**

We've covered a lot of ground so far on the culture and composition of teams writing software, and in this chapter, we'll take a look at the person ultimately responsible for making it all work.

到目前為止，我們已經介紹了編寫軟體團隊的文化和組成，在本章中，我們將看看最終負責使所有工作正常進行的人。

No team can function well without a leader, especially at Google, where engineering is almost exclusively a team endeavor. At Google, we recognize two different leadership roles. A *Manager* is a leader of people, whereas a *Tech Lead* leads technology efforts. Although the responsibilities of these two roles require similar planning skills, they require quite different people skills.

沒有領導者，任何團隊都無法正常運作，尤其是在 Google，工程幾乎完全是團隊的努力。在 Google，我們認識到兩種不同的領導者角色。*經理*是人的領導，而*技術負責人*則領導技術工作。雖然這兩個角色的職責需要類似的規劃技能，但他們需要相當不同的人際交往技能。

A boat without a captain is nothing more than a floating waiting room: unless someone grabs the rudder and starts the engine, it's just going to drift along aimlessly with the current. A piece of software is just like that boat: if no one pilots it, you're left with a group of engineers burning up valuable time, just sitting around waiting for something to happen (or worse, still writing code that you don't need). Although this chapter is about people management and technical leadership, it is still worth a read if you're an individual contributor because it will likely help you understand your own leaders a bit better.

沒有船長的船隻不過是一個漂浮的等候室：除非有人抓住方向舵並啟動引擎，否則它只會隨著水流漫無目的地漂流。一個軟體就像那艘船：如果沒有人駕駛它，你會被一群工程師浪費寶貴的時間，只是坐在那裡等待一些事情發生（或者更糟糕的是，仍然在編寫你不需要的程式碼）。儘管本章是關於人員管理和技術領導的，但如果你是個人貢獻者，則仍值得一讀，因為它可能會幫助你更好地瞭解你是自己的領導。

# Managers and Tech Leads (and Both) 經理和技術負責人（以及兩者）

Whereas every engineering team generally has a leader, they acquire those leaders in different ways. This is certainly true at Google; sometimes an experienced manager comes in to run a team, and sometimes an individual contributor is promoted into a leadership position (usually of a smaller team).

雖然每個工程團隊通常都有一個領導者，但他們以不同的方式獲得這些領導者。這在 Google 肯定是一樣的；有時是一位經驗豐富的經理來管理一個團隊，有時是一位因個人貢獻被提升為領導職位（通常是一個較小的團隊）。

In nascent teams, both roles will sometimes be filled by the same person: a *Tech Lead Manager* (TLM). On larger teams, an experienced people manager will step in to take on the management role while a senior engineer with extensive experience will step into the tech lead role. Even though manager and tech lead each play an important part in the growth and productivity of an engineering team, the people skills required to succeed in each role are wildly different.

在新生團隊中，這兩個角色有時會由同一個人擔任：*技術主管經理*（TLM）。在較大的團隊中，有經驗的人事經理將介入管理角色，而具有豐富經驗的高階工程師將進入技術負責人的角色。儘管經理和技術負責人在工程團隊的成長和生產效率方面都發揮著重要作用，但在每個角色中取得成功所需的人際關係技能卻大不相同。

## The Engineering Manager 工程經理

Many companies bring in trained people managers who might know little to nothing about software engineering to run their engineering teams. Google decided early on, however, that its software engineering managers should have an engineering background. This meant hiring experienced managers who used to be software engineers, or training software engineers to be managers (more on this later).

許多公司引進了訓練有素的人事經理，他們可能對軟體工程知之甚少，甚至一無所知來管理工程團隊。然而，Google 很早就決定，其軟體工程經理應該有工程背景。這意味著僱用曾經是軟體工程師的有經驗的經理，或者培訓軟體工程師成為經理（後面會有更多介紹）。

At the highest level, an engineering manager is responsible for the performance, productivity, and happiness of every person on their team—including their tech lead— while still making sure that the needs of the business are met by the product for which they are responsible. Because the needs of the business and the needs of individual team members don't always align, this can often place a manager in a difficult position.

在最高層面上，工程經理要對其團隊中每個人的績效、生產效率和幸福感負責——包括他們的技術負責人——同時還要確保他們負責的產品能夠滿足企業的需求。由於企業的需求和個人團隊成員的需求並不總是一致，這往往會使經理處於困境。

## The Tech Lead 技術負責人

The tech lead (TL) of a team—who will often report to the manager of that team—is responsible for (surprise!) the technical aspects of the product, including technology decisions and choices, architecture, priorities, velocity, and general project management (although on larger teams they might have program managers helping out with this). The TL will usually work hand in hand with the engineering manager to ensure that the team is adequately staffed for their product and that engineers are set to work on tasks that best match their skill sets and skill levels. Most TLs are also individual contributors, which often forces them to choose between doing something quickly themselves or delegating it to a team member to do (sometimes) more slowly. The latter is most often the correct decision for the TL as they grow the size and capability of their team.

團隊的技術負責人（TL）通常會向該團隊的經理彙報，負責產品的技術方面，包括技術決策和選擇、架構、優先順序、速度和總體專案管理（儘管在較大的團隊中，他們可能會有專案經理幫助處理這個問題）。TL 通常會與工程經理攜手合作，以確保團隊有足夠的人員來完成他們的產品，並確保工程師被安排在最符合他們技能組合和技能水平的任務上工作。大多數 TL 也是個人貢獻者，這往往迫使他們在自己快速做某事或委託團隊成員做（有時）更慢的事之間做出選擇。對於 TL 來說，隨著團隊規模和能力的增長，後者通常是正確的決策。

## The Tech Lead Manager 技術主管經理

On small and nascent teams for which engineering managers need a strong technical skill set, the default is often to have a TLM: a single person who can handle both the people and technical needs of their team. Sometimes, a TLM is a more senior person, but more often than not, the role is taken on by someone who was, until recently, an individual contributor.

在小型和新生的團隊中，工程經理需要強大的技術技能，預設情況下，通常會有一個 TLM（技術主管經理）：一個可以同時處理團隊的人員和技術需求的人。有時，TLM（技術主管經理）是一個更進階的人，但更多的時候，這個角色是由一個直到最近還是個人貢獻者的人承擔的。

At Google, it's customary for larger, well-established teams to have a pair of leaders— one TL and one engineering manager—working together as partners. The theory is that it's really difficult to do both jobs at the same time (well) without completely burning out, so it's better to have two specialists crushing each role with dedicated focus.

在 Google，大型、成熟的團隊習慣於有一對領導——一個 TL（技術負責人）和一個工程經理——作為夥伴一起工作。理論上說，要同時做這兩份工作（好）而又不至於完全精疲力竭，真的很困難，因此最好有兩名專家專注於每個角色。

The job of TLM is a tricky one and often requires the TLM to learn how to balance individual work, delegation, and people management. As such, it usually requires a high degree of mentoring and assistance from more experienced TLMs. (In fact, we recommend that in addition to taking a number of classes that Google offers on this subject, a newly minted TLM seek out a senior mentor who can advise them regularly as they grow into the role.)

TLM（技術主管經理）的工作很棘手，通常需要 TLM 學習如何平衡個人工作、授權和人員管理。因此，它通常需要經驗豐富的 TLM 的深入指導和幫助。（事實上，我們建議新晉的 TLM 除了學習 Google 提供的一些關於這一主題的課程外，還應該尋找一位資深導師，在他們成長為該角色時定期向他們提供建議。）

### Case Study: Influencing Without Authority 案例研究：沒有權威的影響力

It's generally accepted that you can get folks who report to you to do the work that you need done for your products, but it's different when you need to get people outside of your organization—or heck, even outside of your product area sometimes—to do something that you think needs to be done. This "influence without authority" is one of the most powerful leadership traits that you can develop.

人們普遍認為，你可以讓向你彙報工作的人為你的產品做你需要做的工作，但當你需要讓你的組織以外的人——或者說，有時甚至是你的產品領域以外的人——做你認為需要做的事情時，情況就不同了。這種"沒有權威的影響力"是你可以培養的最強大的領導力特質之一。

For example, for years, Jeff Dean, senior engineering fellow and possibly the most well-known Googler *inside* of Google, led only a fraction of Google's engineering team, but his influence on technical decisions and direction reaches to the ends of the entire engineering organization and beyond (thanks to his writing and speaking outside of the company).

例如，多年來，高階工程研究員傑夫·迪安（Jeff Dean）可能是 Google 內部最知名的 Googler，他只領導 Google 工程團隊的一小部分，但他對技術決策和方向的影響卻達到了整個工程組織的末端，甚至更遠（這要歸功於他在公司之外的寫作和演講）。

Another example is a team that I started called The Data Liberation Front: with a team of less than a half-dozen engineers, we managed to get more than 50 Google products to export their data through a product that we launched called Google Takeout. At the time, there was no formal directive from the executive level at Google for all products to be a part of Takeout, so how did we get hundreds of engineers to contribute to this effort? By identifying a strategic need for the company, showing how it linked to the mission and existing priorities of the company, and working with a small group of engineers to develop a tool that allowed teams to quickly and easily integrate with Takeout.

另一個例子是我發起的一個名為"資料解放陣線"的團隊：一個由不到六名工程師組成的團隊，我們成功地讓 50 多個 Google 產品透過我們推出的一款名為"Google Takeout"的產品來輸出資料。當時，Google 的管理層並沒有正式指示所有產品都要成為 Takeout 的一部分，那麼我們是如何讓數百名工程師為這項工作做出貢獻的呢？透過確定公司的戰略需求，展示它如何與公司的使命和現有的優先事項相聯絡，並與一小組工程師合作開發一個工具，使團隊能夠快速、輕鬆地與 Takeout 整合。

---

## Moving from an Individual Contributor Role to a Leadership Role 從個人貢獻者角色轉變為領導角色

Whether or not they're officially appointed, someone needs to get into the driver's seat if your product is ever going to go anywhere, and if you're the motivated, impatient type, that person might be you. You might find yourself sucked into helping your team resolve conflicts, make decisions, and coordinate people. It happens all the time, and often by accident. Maybe you never intended to become a "leader," but somehow it happened anyway. Some people refer to this affliction as "manageritis."

無論他們是否被正式任命，如果你的產品要發展，就需要有人進入駕駛座，如果你是一個有動力、有影響力的人，那麼這個人可能就是你。你可能會發現自己主動去幫助你的團隊解決衝突、做出決策和協調人員的行列。這種情況一直在存在，而且往往是偶然發生的。也許你從來沒有想過要成為一個 "領導者"，但不知何故，它還是發生了。有些人把這種痛苦稱為 "經理病"。

Even if you've sworn to yourself that you'll never become a manager, at some point in your career, you're likely to find yourself in a leadership position, especially if you've been successful in your role. The rest of this chapter is intended to help you understand what to do when this happens.

即便你向自己發誓你永遠不會成為一名管理者，在你職業生涯的某個階段，你很可能會發現自己處於領導位置，特別是如果你在自己的角色上取得了成功。本章的其餘部分旨在幫助你理解當這種情況發生時該怎麼做。

We're not here to attempt to convince you to become a manager, but rather to help show why the best leaders work to serve their team using the principles of humility, respect, and trust. Understanding the ins and outs of leadership is a vital skill for influencing the direction of your work. If you want to steer the boat for your project and not just go along for the ride, you need to know how to navigate, or you'll run yourself (and your project) onto a sandbar.

我們來這裡不是為了說服你成為一名管理者，而是為了幫助你展示為什麼最好的領導者會以謙遜、尊重和信任的原則為團隊服務。瞭解領導力的來龍去脈是影響你工作方向的重要技能。如果你想為你的專案掌舵，而不是隨波逐流，你需要知道如何導航，否則你會把自己（和你的專案）撞上沙灘。

## The Only Thing to Fear Is…Well, Everything 唯一害怕的是…嗯，所有的事

Aside from the general sense of malaise that most people feel when they hear the word "manager," there are a number of reasons that most people don't want to become managers. The biggest reason you'll hear in the software development world is that you spend much less time writing code. This is true whether you become a TL or an engineering manager, and I'll talk more about this later in the chapter, but first, let's cover some more reasons why most of us avoid becoming managers.

除了大多數人聽到"經理"這個詞時普遍感到的不安之外，大多數人不想成為經理的原因有很多。在軟體開發領域，你會聽到的最大的原因是你花在編寫程式碼上的時間要少得多。無論你是成為 TL 還是工程經理，這都是事實，我將在本章後面更多地討論這一點，但首先，讓我們討論一下更多的原因，為什麼我們大多數人都逃避成為經理。

If you've spent the majority of your career writing code, you typically end a day with something you can point to—whether it's code, a design document, or a pile of bugs you just closed—and say, "That's what I did today." But at the end of a busy day of "management," you'll usually find yourself thinking, "I didn't do a damned thing today." It's the equivalent of spending years counting the number of apples you picked each day and changing to a job growing bananas, only to say to yourself at the end of each day, "I didn't pick any apples," happily ignoring the flourishing banana trees sitting next to you. Quantifying management work is more difficult than counting widgets you turned out, but just making it possible for your team to be happy and productive is a big measure of your job. Just don't fall into the trap of counting apples when you're growing bananas.[1]

如果你的職業生涯大部分時間都在寫程式碼，你通常會在結束一天的工作時，指著一些東西——無論是程式碼、設計文件，還是你剛剛關閉的一堆 bug——說："這就是我今天做的事情。" 但在忙碌的 "管理 "一天結束時，你通常會發現自己在想，"我今天一件該死的事也沒做"。這相當於花上幾年的時間數數你每天摘的蘋果的數量，然後換了份種植香蕉的工作，每天結束時都對自己說，"我沒有摘蘋果"，忽略了你身旁茂盛的香蕉樹。量化管理工作比計算你生產的小元件要困難得多，但讓你的團隊能夠快樂且高效是你工作的一個重要指標。只是在你種植香蕉時不要落入數蘋果的陷阱。

Another big reason for not becoming a manager is often unspoken but rooted in the famous “Peter Principle,” which states that “In a hierarchy every employee tends to rise to his level of incompetence.” Google generally avoids this by requiring that a person perform the job *above* their current level for a period of time (i.e., to “exceeds expectations” at their current level) before being promoted to that level. Most people have had a manager who was incapable of doing their job or was just really bad at managing people,[2] and we know some people who have worked only for bad managers. If you’ve been exposed only to crappy managers for your entire career, why would you *ever* want to be a manager? Why would you want to be promoted to a role that you don’t feel able to do?

不成為經理的另一個重要原因往往是不言而喻的，但卻源於著名的 "彼得原理"，即 "在等級制度中，每個職工趨向於上升到他所不能勝任的職位"。Google 通常透過要求一個人在晉升到該級別之前，必須在其目前的級別上完成高於當前水平的工作一段時間（即達到 "超出預期 "的目前級別）來避免這種情況。大多數人都有過一個不能勝任工作的經理，或者只是非常不善於管理人，而我們知道有些人只為糟糕的經理工作。如果你在整個職業生涯中只接觸過糟糕的經理，你為什麼**想成為一名經理**？你為什麼要被提拔到一個你覺得無法勝任的角色？

> 彼得原理推導：每一個職位最終都將被一個不能勝任其工作的職工所佔據。層級組織的工作任務多半是由尚未達到勝任階層的員工完成的。

There are, however, great reasons to consider becoming a TL or manager. First, it’s a way to scale yourself. Even if you’re great at writing code, there’s still an upper limit to the amount of code you can write. Imagine how much code a team of great engineers could write under your leadership! Second, you might just be really good at it—many people who find themselves sucked into the leadership vacuum of a project discover that they’re exceptionally skilled at providing the kind of guidance, help, and air cover a team or a company needs. Someone has to lead, so why not you?

然而，有很好的理由考慮成為一名 TL 或經理。首先，這是一種擴充自己的方式。即使你很擅長寫程式碼，你能寫的程式碼量還是有上限的。想象一下，在你的領導下，一個由優秀工程師組成的團隊可以寫多少程式碼？第二，你可能真的很擅長寫程式碼——許多人發現自己被吸進了專案的領導真空中，發現自己在提供團隊或公司所需的指導、幫助和故障處理方面非常熟練。總得有人來領導，那麼為什麼不是你呢？

> 1 另一個需要適應的差異是，我們作為管理者所做的事情通常會在更長的時間後才得到回報。
>
> 2 然而，另一個原因是，公司不應該強迫人們把管理作為職業道路的一部分：如果一個工程師能夠寫出大量的優秀程式碼，卻完全沒有管理人或領導團隊的願望，強迫他們進入管理層或 TL 的角色，你就會失去一個偉大的工程師，得到一個糟糕的經理。這不僅是一個糟糕的主意，而且是積極有害的方式。

## Servant Leadership 服務型領導

There seems to be a sort of disease that strikes managers in which they forget about all the awful things their managers did to them and suddenly begin doing these same things to “manage” the people that report to them. The symptoms of this disease include, but are by no means limited to, micromanaging, ignoring low performers, and hiring pushovers. Without prompt treatment, this disease can kill an entire team. The best advice I received when I first became a manager at Google was from Steve Vinter, an engineering director at the time. He said, “Above all, resist the urge to manage.” One of the greatest urges of the

newly minted manager is to actively “manage” their employees because that’s what a manager does, right? This typically has disastrous consequences.

似乎有一種疾病襲擾了經理們，他們忘記了他們的經理對他們所做的所有可怕的事情，突然開始做同樣的事情來 "管理 "向他們彙報的人。這種疾病的症狀包括，但不限於，微觀管理（事必躬親），忽視低績效員工，以及使用推卸責任者。如果不及時治療，這種疾病可以殺死整個團隊。當我第一次在 Google 成為經理時，我得到的最好的建議是來自當時的工程總監史蒂夫·溫特。他說：“首先，要抵制管人的衝動"。新上任的經理人最大的衝動之一就是積極 "管理 "他們的員工，因為這就是經理的工作，對嗎？這通常會帶來災難性的後果。

The cure for the “management” disease is a liberal application of “servant leadership,” which is a nice way of saying the most important thing you can do as a leader is to serve your team, much like a butler or majordomo tends to the health and well-being of a household. As a servant leader, you should strive to create an atmosphere of humility, respect, and trust. This might mean removing bureaucratic obstacles that a team member can’t remove by themselves, helping a team achieve consensus, or even buying dinner for the team when they’re working late at the office. The servant leader fills in the cracks to smooth the way for their team and advises them when necessary, but still isn’t afraid of getting their hands dirty. The only managing that a servant leader does is to manage both the technical and social health of the team; as tempting as it might be to focus on purely the technical health of the team, the social health of the team is just as important (but often infinitely more difficult to manage).

“管理”疾病的治療方法是“服務型領導”的自由運用這是一個很好的做法，作為一個領導者，你能做的最重要的事情就是為你的團隊服務，就像一個管家或大管家關心一個家庭的健康和福祉一樣。作為一個服務型領導者，你應該努力營造一種謙遜、尊重和信任的氛圍。這可能意味著消除團隊成員自己無法消除的官僚主義障礙，幫助團隊達成共識，甚至在團隊在辦公室工作到很晚的時候為他們買晚餐。服務型領導會填補縫隙，為他們的團隊鋪平道路，必要時為他們提供建議，但仍然不怕弄髒自己的手。服務型領導所做的唯一管理就是管理團隊的技術和社會健康；儘管單純關注團隊的技術健康可能很誘人，但團隊的氛圍健康也同樣重要（但往往更難管理）。

# The Engineering Manager 工程經理

So, what is actually expected of a manager at a modern software company? Before the computing age, “management” and “labor” might have taken on almost antagonistic roles, with the manager wielding all of the power and labor requiring collective action to achieve its own ends. But that isn’t how modern software companies work.

那麼，在現代軟體公司中，對經理的實際期望是什麼？在計算機時代之前，"管理 "和 "勞動 "可能承擔著幾乎是對立的角色，管理者掌握著所有的權力，而勞動者則需要集體行動來實現自己的目的。但這並不是現代軟體公司的工作方式。

## Manager Is a Four-Letter Word 經理是一個由四個字母組成的詞

Before talking about the core responsibilities of an engineering manager at Google, let’s review the history of managers. The present-day concept of the pointy-haired manager is partially a carryover, first from military hierarchy and later adopted by the Industrial Revolution—more than a hundred years ago! Factories began popping up everywhere, and they required (usually unskilled) workers to keep the machines going. Consequently, these workers required supervisors to manage them, and because it was easy to replace these workers with other people who were desperate for a job, the managers had little motivation to treat their employees well or improve conditions for them. Whether humane or not, this method worked well for many years when the employees had nothing more to do than perform rote tasks.

在談論 Google 工程經理的核心職責之前，讓我們回顧一下經理的歷史。今天的頂尖經理的概念部分是延續下來的，首先是來自軍隊的等級制度，後來被工業革命所採用——一百多年前！工廠開始到處湧現，他們需要（通常是不熟練的）工人來維持機器運轉。因此，這些工人需要主管人員來管理他們，由於很容易用其他急於找工作的人取代這些工人，主管人員沒有什麼動力來善待他們的僱員或改善他們的條件。無論人道與否，這種方法在許多年裡都很有效，當時員工除了完成死記硬背的任務外沒有其他事情可做。

Managers frequently treated employees in the same way that cart drivers would treat their mules: they motivated them by alternately leading them forward with a carrot, and, when that didn't work, whipping them with a stick. This carrot-and-stick method of management survived the transition from the factory [3] to the modern office, where the stereotype of the tough-as-nails manager-as-mule-driver flourished in the middle part of the twentieth century when employees would work at the same job for years and years.

經理們經常用趕車人對待騾子的方式來對待員工：他們用胡蘿蔔來激勵他們，當胡蘿蔔不起作用的時候，就用棍子來鞭打他們。這種胡蘿蔔加大棒的管理方法在從工廠過渡到現代辦公室的過渡中倖存下來，在二十世紀中葉，當員工在同一工作崗位上工作多年後，強硬的經理人作為騾子趕車人的刻板印象在辦公室裡盛行。

This continues today in some industries—even in industries that require creative thinking and problem solving—despite numerous studies suggesting that the anachronistic carrot and stick is ineffective and harmful to the productivity of creative people. Whereas the assembly-line worker of years past could be trained in days and replaced at will, software engineers working on large codebases can take months to get up to speed on a new team. Unlike the replaceable assembly-line worker, these people need nurturing, time, and space to think and create.

儘管許多研究表明，不合時宜的胡蘿蔔和大棒是無效的，而且對有創造力的人的生產效率有害，但這種情況今天在一些行業仍然存在——甚至在那些需要創造力思維和解決問題的行業。過去幾年的裝配線工人可以在幾天內接受培訓並被隨意替換，而從事大型程式碼庫工作的軟體工程師可能需要幾個月的時間才能在一個新的團隊中適應。與可替換的裝配線工人不同，這些人需要培養、時間和空間來思考和創造。

> 3 有關優化工廠工人流動的更多有趣資訊，請閱讀科學管理或泰勒主義，尤其是其對工人士氣的影響。

## Today's Engineering Manager 當今的工程經理

Most people still use the title "manager" despite the fact that it's often an anachronism. The title itself often encourages new managers to *manage* their reports. Managers can wind up acting like parents, [4] and consequently employees react like children. To frame this in the context of humility, respect, and trust: if a manager makes it obvious that they trust their employee, the employee feels positive pressure to live up to that trust. It's that simple. A good manager forges the way for a team, looking out for their safety and well-being, all while making sure their needs are met. If there's one thing you remember from this chapter, make it this:
Traditional managers worry about how to get things done, whereas great managers worry about what things get done (and trust their team to figure out how to do it).

大多數人仍然使用 "經理 "這個頭銜，儘管它往往是一個不合時宜的頭銜。這個頭銜本身經常鼓勵新經理撰寫報告。經理們可能會表現得像父母一樣，因此員工的反應也像孩子。從謙遜、尊重和信任的角度來看：如果經理明顯地表示他們信任員工，那麼員工就會感到有積極的壓力，不辜負這種信任。就是這麼簡單。好的經理為團隊開闢道路，關注他們的安全和福祉，同時確保他們的需求得到滿足。如果你在本章中記住了一件事，那就是這個：
傳統的經理擔心的是如何把事情做好，而優秀的經理擔心的是把什麼事情做好（並相信他們的團隊能想出辦法來）。

A new engineer, Jerry, joined my team a few years ago. Jerry's last manager (at a different company) was adamant that he be at his desk from 9:00 to 5:00 every day, and assumed that if he wasn't there, he wasn't working enough (which is, of course, a ridiculous assumption). On his first day working with me, Jerry came to me at 4:40p.m. and stammered out an apology that he had to leave 15 minutes early because he had an appointment that he had been unable to reschedule. I looked at him, smiled, and told him flat out, "Look, as long as you get your job done, I don't care what time you leave the office." Jerry stared blankly at me for a few seconds, nodded, and went on his way. I treated Jerry like an adult; he always got his work done, and I never had to worry about him being at his desk, because he didn't need a babysitter to get his work done. If your employees are so uninterested in their job that they actually need traditional-manager babysitting to be convinced to work, *that* is your real problem.

幾年前，一位名叫傑瑞的工程師新加入了我的團隊。傑瑞的上一任經理（在另一家公司）堅決要求他每天從早上 9 點到下午 5 點都在辦公桌前，並認為如果他不在那裡，就說明他工作得不夠努力（當然，這是一個可笑的假設）。在他和我一起工作的第一天，傑瑞在下午 4 點 40 分來找我，結結巴巴地道歉，說他不得不提前 15 分鐘離開，因為他有一個約會，但他沒法重新安排。我看著他，笑了笑，直截了當地告訴他："聽著，只要你完成了你的工作，我不在乎你什麼時候離開辦公室。" 傑瑞茫然地盯著我看了幾秒鐘，點了點頭，然後上路了。我對待傑瑞就像對待成年人一樣，他總是能完成工作，我從來沒有擔心過他是否在辦公桌前，因為他不需要保姆一樣的來完成工作。如果你的員工對他們的工作如此不感興趣，以至於他們實際上需要傳統的經理保姆式來監督他們工作，這才是你真正的問題。

> 4 如果你有孩子，你很有可能清楚地記得你第一次對你的孩子說了什麼，讓你停下來並感嘆（也許甚至是大聲地感嘆）："我的媽呀，我已經變成我的母親了。"

---

## Failure Is an Option 失敗也是一種選擇

Another way to catalyze your team is to make them feel safe and secure so that they can take greater risks by building psychological safety—meaning that your team members feel like they can be themselves without fear of negative repercussions from you or their team members. Risk is a fascinating thing; most humans are terrible at evaluating risk, and most companies try to avoid risk at all costs. As a result, the usual modus operandi is to work conservatively and focus on smaller successes, even when taking a bigger risk might mean exponentially greater success. A common saying at Google is that if you try to achieve an impossible goal, there's a good chance you'll fail, but if you fail trying to achieve the impossible, you'll most likely accomplish far more than you would have accomplished had you merely attempted something you knew you could complete. A good way to build a culture in which risk taking is accepted is to let your team know that it's OK to fail.

催化你的團隊的另一個方法是讓他們感到安全和有保障，這樣他們就可以透過建立心理安全來承擔更大的風險——也就是說，你的團隊成員覺得他們可以做自己，而不用擔心來自你或團隊成員的負面反饋。風險是一個令人著迷的東西；大多數人在評估風險方面是很糟糕的，而且大多數公司試圖不惜一切代價避免風險。因此，通常的工作方式是保守地工作，專注於較小的成功，即使承擔較大的風險可能意味著成倍的成功。在 Google，有一句話是這樣說的：如果你試圖實現一個不可能的目標，你很有可能會

失敗，但如果你試圖實現不可能的目標而失敗，你的成就很可能會遠遠超過你僅僅嘗試你知道你可以完成的事情所取得的成就。建立一個接受風險的文化的好方法是讓你的團隊知道，失敗是允許的。

So, let's get that out of the way: it's OK to fail. In fact, we like to think of failure as a way of learning a lot really quickly (provided that you're not repeatedly failing at the same thing). In addition, it's important to see failure as an opportunity to learn and not to point fingers or assign blame. Failing fast is good because there's not a lot at stake. Failing slowly can also teach a valuable lesson, but it is more painful because more is at risk and more can be lost (usually engineering time). Failing in a manner that affects customers is probably the least desirable failure that we encounter, but it's also one in which we have the greatest amount of structure in place to learn from failures. As mentioned earlier, every time there is a major production failure at Google, we perform a postmortem. This procedure is a way to document the events that led to the actual failure and to develop a series of steps that will prevent it from happening in the future. This is neither an opportunity to point fingers, nor is it intended to introduce unnecessary bureaucratic checks; rather, the goal is to strongly focus on the core of the problem and fix it once and for all. It's very difficult, but quite effective (and cathartic).

所以，讓我們先把話說清楚：失敗是允許的。事實上，我們喜歡把失敗看作是一種快速學習的方式（只要你不是在同一件事上反覆犯錯）。此外，重要的是把失敗看作是一個學習的機會，而不是指責或推責。快速失敗是好的，因為沒有太多的風險。緩慢的失敗也能給人以寶貴的教訓，但它更痛苦，因為風險更大，可能損失更多（通常是工程時間）。以影響客戶的方式失敗可能是我們遇到的最不可取的失敗，但這也是我們從失敗中學習的最大方式。如前所述，每次 Google 出現重大生產故障時，我們都會進行事後分析。這一過程是記錄失敗的一種方式導致實際失敗的事件，並制定一系列措施防止未來發生。這既不是一個指責的機會，也不是為了引入不必要的官僚檢查；相反，目標是強烈關注問題的核心，並一勞永逸地解決它。這非常困難，但相當有效（和宣洩）。

Individual successes and failures are a bit different. It's one thing to laud individual successes, but looking to assign individual blame in the case of failure is a great way to divide a team and discourage risk taking across the board. It's alright to fail, but fail as a team and learn from your failures. If an individual succeeds, praise them in front of the team. If an individual fails, give constructive criticism in private.[5] Whatever the case, take advantage of the opportunity and apply a liberal helping of humility, respect, and trust to help your team learn from its failures.

個人的成功和失敗是有點不同的。讚揚個人的成功是一回事，但在失敗的情況下尋找個人的責任是分裂團隊的方式，並阻礙整個團隊的風險承擔。失敗是正常的，但作為一個團隊，要從失敗中學習。如果一個人成功了，在團隊面前表揚他們。如果一個人失敗了，私下給予建設性的指導。無論是什麼情況，都要利用這個機會，運用謙遜、尊重和信任的慷慨幫助，幫助你的團隊從失敗中吸取教訓。

---

> 5 對一個人的公開批評不僅沒有效果（它使人們處於防禦狀態），而且很少有必要，大多數時候只是刻薄或殘酷。你可以肯定，團隊的其他成員已經知道一個人什麼時候失敗了，所以沒有必要重複。

# Antipatterns 反模式

Before we go over a litany of “design patterns” for successful TLs and engineering managers, we’re going to review a collection of the patterns that you *don’t* want to follow if you want to be a successful manager. We’ve observed these destructive patterns in a handful of bad managers that we’ve encountered in our careers, and in more than a few cases, ourselves.

在我們討論一系列成功的 TL 和工程經理的 "設計模式 "之前，我們要回顧一下，如果你想成為一個成功的經理，你*不能*遵循的模式。我們已經在我們職業生涯中遇到的一些糟糕經理身上觀察到了這些破壞性的模式，而我們自己也遇到過不少這樣的情況。

## Antipattern: Hire Pushovers 反模式：僱傭推手

If you’re a manager and you’re feeling insecure in your role (for whatever reason), one way to make sure no one questions your authority or threatens your job is to hire people you can push around. You can achieve this by hiring people who aren’t as smart or ambitious as you are, or just people who are more insecure than you. Even though this will cement your position as the team leader and decision maker, it will mean a lot more work for you. Your team won’t be able to make a move without you leading them like dogs on a leash. If you build a team of pushovers, you probably can’t take a vacation; the moment you leave the room, productivity comes to a screeching halt. But surely this is a small price to pay for feeling secure in your job, right?

如果你是一名經理，而且你對自己的角色感到沒安全感（無論什麼原因），確保沒有人質疑你的權威或威脅你的工作的一個方法是僱傭你可以推波助瀾的人。你可以透過僱傭不如你聰明或有野心的人，或者只是比你更沒有安全感的人，來實現這一目標。儘管這將鞏固你作為團隊領導者和決策者的地位，但這將意味著你的工作會更多。如果你不像牽著狗一樣牽著他們，你的團隊就無法行動。如果你建立了一個由推手組成的團隊，你可能就不能休假了；你離開房間的那一刻，工作效率就急劇下降。但是，這肯定是為你的工作安全感付出的一個小代價，對嗎？

Instead, you should strive to hire people who are smarter than you and can replace you. This can be difficult because these very same people will challenge you on a regular basis (in addition to letting you know when you make a mistake). These very same people will also consistently impress you and make great things happen. They’ll be able to direct themselves to a much greater extent, and some will be eager to lead the team, as well. You shouldn’t see this as an attempt to usurp your power; instead, look at it as an opportunity for you to lead an additional team, investigate new opportunities, or even take a vacation without worrying about checking in on the team every day to make sure it’s getting its work done. It’s also a great chance to learn and grow—it’s a lot easier to expand your expertise when surrounded by people who are smarter than you.

相反，你應該努力僱用比你更聰明的人，並能取代你。這可能很困難，因為這些人也會定期挑戰你（除了在你犯錯時只有你自己知道）。這些人也會不斷地給你留下深刻印象，並讓偉大的事情出現。他們將能夠在更大程度上指導自己，有些人也會渴望領導團隊。你不應該把這看作是企圖篡奪你的權力；相反，你應該把它看作是一個機會，讓你領導一個額外的團隊，找到新的機會，甚至休假，而不必擔心每天都要檢查團隊，確保它完成工作。這也是一個學習和成長的好機會——當週圍有比你更聰明的人時，拓展你的專業知識會容易得多。

## Antipattern: Ignore Low Performers 反模式：忽略低績效員工

Early in my career as a manager at Google, the time came for me to hand out bonus letters to my team, and I grinned as I told my manager, “I love being a manager!” Without missing a beat, my manager, a long-time industry veteran, replied, “Sometimes you get to be the tooth fairy, other times you have to be the dentist.”

在我在 Google 擔任經理的早期，到了給我的團隊發獎金信的時候，我咧嘴笑著對我的經理說："我喜歡當經理！" 我的經理是一位長期的行業老手，他不慌不忙地回答說："有時你可以做牙仙，有時你必須做牙醫。"

It's never any fun to pull teeth. We've seen team leaders do all the right things to build incredibly strong teams only to have these teams fail to excel (and eventually fall apart) because of just one or two low performers. We understand that the human aspect is the most challenging part of writing software, but the most difficult part of dealing with humans is handling someone who isn't meeting expectations. Sometimes, people miss expectations because they're not working long enough or hard enough, but the most difficult cases are when someone just isn't capable of doing their job no matter how long or hard they work.

拔牙從來不是什麼有趣的事情。我們看到團隊領導做了所有正確的事情，建立了令人難以置信的強大團隊，但這些團隊卻因為一兩個表現不佳的人而無法脫穎而出（並最終分崩離析）。我們知道，有些人在編寫軟體最具挑戰性的部分是出色的，但與人打交道最困難的部分是處理沒有達到預期的效果。有時，人們達不到預期是因為他們工作的時間不夠長或不夠努力，但最困難的情況是有人無論工作多長時間或多努力，就是沒有能力完成他們的工作。

Google's Site Reliability Engineering (SRE) team has a motto: "Hope is not a strategy." And nowhere is hope more overused as a strategy than in dealing with a low performer. Most team leaders grit their teeth, avert their eyes, and just *hope* that the low performer either magically improves or just goes away. Yet it is extremely rare that this person does either.

Google 的網站可靠性工程（SRE）團隊有一個座右銘："希望不是一種策略"。而希望作為一種策略被過度使用的情況，莫過於在處理低績效員工時。大多數團隊領導咬緊牙關，轉移視線，只是*希望*低績效員工要麼神奇地改善，要麼就消失。然而，這種情況極少發生。

While the leader is hoping and the low performer isn't improving (or leaving), high performers on the team waste valuable time pulling the low performer along, and team morale leaks away into the ether. You can be sure that the team knows the low performer is there even if you're ignoring them—in fact, the team is *acutely* aware of who the low performers are, because they have to carry them.

當領導抱著希望，而低績效者沒有進步（或離開）的時候，團隊中高績效者就會浪費寶貴的時間來拉攏低績效員工，而團隊就會洩氣。你可以肯定的是，即使你對他們忽視，團隊也知道低績效員工的存在——事實上，團隊非常清楚誰是低績效員工，因為他們必須要帶著他們。

Ignoring low performers is not only a way to keep new high performers from joining your team, but it's also a way to encourage existing high performers to leave. You eventually wind up with an entire team of low performers because they're the only ones who can't leave of their own volition. Lastly, you aren't even doing the low performer any favors by keeping them on the team; often, someone who wouldn't do well on your team could actually have plenty of impact somewhere else.

忽視低績效員工不僅會阻礙新的高績效員工加入你的團隊，而且也會鼓勵現有的高績效員工離開。你最終會發現整個團隊都是低績效員工，因為他們是唯一不能主動離開的人。最後，你把低績效員工留在團隊中，甚至對他們沒有任何好處；通常情況下，一個在你的團隊中做得不好的人，實際上可以在其他地方產生很大的影響。

The benefit of dealing with a low performer as quickly as possible is that you can put yourself in the position of helping them up or out. If you immediately deal with a low performer, you'll often find that they merely need some encouragement or direction to slip into a higher state of productivity. If you wait too long to deal with a low performer, their relationship with the team is going to be so sour and you're going to be so frustrated that you're not going to be able to help them.

儘快處理低績效員工的好處是，你可以把自己放在幫助他們提升或退出的位置上。如果你立即處理一個低績效員工，你往往會發現他們只需要一些鼓勵或指導就能走向更高的生產效率狀態。如果你等待很長時間再來處理低績效員工，他們與團隊的關係就會變得非常糟糕，你也會感到非常沮喪，以至於你無法幫助他們。

How do you effectively coach a low performer? The best analogy is to imagine that you're helping a limping person learn to walk again, then jog, then run alongside the rest of the team. It almost always requires temporary micromanagement, but still a whole lot of humility, respect, and trust—particularly respect. Set up a specific time frame (say, two months) and some very specific goals you expect them to achieve in that period. Make the goals small, incremental, and measurable so that there's an opportunity for lots of small successes. Meet with the team member every week to check on progress, and be sure you set really explicit expectations around each upcoming milestone so that it's easy to measure success or failure. If the low performer can't keep up, it will become quite obvious to both of you early in the process. At this point, the person will often acknowledge that things aren't going well and decide to quit; in other cases, determination will kick in and they'll "up their game" to meet expectations. Either way, by working directly with the low performer, you're catalyzing important and necessary changes.

你如何有效地指導低績效員工？最好的比喻是想象你在幫助一個跛腳的人重新學會走路，然後慢跑，然後和團隊的其他成員一起跑步。這幾乎總是需要暫時的微觀管理，但仍然需要大量的謙遜、尊重和信任——特別是尊重。設定一個具體的時間框架（例如，兩個月）和一些非常具體的目標，你希望他們在這段時間內實現。把目標定得小一點，漸進一點，可衡量一點，這樣就有機會取得很多小的成功。每週與團隊成員見面，檢查進展情況，並確保你對每個即將到來的里程碑設定了非常明確的期望，這樣就很容易衡量成功或失敗。如果表現不佳的人跟不上，那麼在這個過程的早期，你們兩個人都會很清楚。在這一點上，這個人通常會承認事情進展不順利，並決定退出；在其他情況下，決心會啟動，他們會 "提高自己的水平"，以滿足期望。無論是哪種情況，透過直接與低績效員工合作，你都會促成重要而必要的改變。

## Antipattern: Ignore Human Issues 反模式：忽視人性的問題

A manager has two major areas of focus for their team: the social and the technical. It's rather common for managers to be stronger in the technical side at Google, and because most managers are promoted from a technical job (for which the primary goal of their job was to solve technical problems), they can tend to ignore human issues. It's tempting to focus all of your energy on the technical side of your team because, as an individual contributor, you spend the vast majority of your time solving technical problems. When you were a student, your classes were all about learning the technical ins and outs of your work. Now that you're a manager, however, you ignore the human element of your team at your own peril.

經理對他們的團隊有兩個主要的關注領域：社會和技術。在 Google，經理在技術方面比較強大是比較常見的，因為大多數經理都是從技術工作中晉升的（對他們來說，工作的主要目標是解決技術問題），他們可能傾向於忽視人性問題。把所有的精力都集中在團隊的技術方面是很誘人的，因為作為個人貢獻者，你的絕大部分時間都在解決技術問題。當你還是個學生的時候，你的課程都是關於學習工作的技術內涵和外延。然而，現在你是一名經理，你忽視了團隊中的人性因素，這將是你自己的危險。

Let's begin with an example of a leader ignoring the human element in his team. Years ago, Jake had his first child. Jake and Katie had worked together for years, both remotely and in the same office, so in the weeks following the arrival of the new baby, Jake worked from home. This worked out great for the couple, and Katie was totally fine with it because she was already used to working remotely with Jake. They were their usual productive selves until their manager, Pablo (who worked in a different office), found out that Jake was working from home for most of the week. Pablo was upset that Jake wasn't going into the office to work with Katie, despite the fact that Jake was just as productive as always and that Katie was fine with the situation. Jake attempted to explain to Pablo that he was just as productive as he would be if he came into the office and that it was much easier on him and his wife for him to mostly work from home for a few weeks. Pablo's response: "Dude, people have kids all the time. You need to go into the office." Needless to say, Jake (normally a mild-mannered engineer) was enraged and lost a lot of respect for Pablo.

讓我們從一個領導忽視其團隊中人性的因素的例子開始。幾年前，傑克有了他的第一個孩子。傑克和凱蒂在一起工作了好幾年，無論是遠端工作還是在同一個辦公室，所以在新生嬰兒出生後的幾周裡，傑克在家工作。這對這對夫婦來說是很好的，凱蒂也完全同意，因為她已經習慣了與傑克一起遠端工作。他們像往常一樣富有成效，直到他們的經理帕布羅（在另一個辦公室工作）發現傑克這周大部分時間都在家裡工作。帕布羅對傑克不到辦公室和凱蒂一起工作感到很不高興，儘管事實上傑克和以前一樣富有成效，而且凱蒂對這種情況也感到滿意。傑克試圖向帕布羅解釋說，如果他進辦公室的話，他的工作效率也是一樣的，而且這幾個星期他大部分時間都在家裡工作，對他和他的妻子來說要容易得多。帕布羅的回答是。"夥計，人們總是有孩子。你需要到辦公室去。" 不用說，傑克（通常是一個溫和的工程師）被激怒了，對帕布羅失去了以往很多尊重。

There are numerous ways in which Pablo could have handled this differently: he could have showed some understanding that Jake wanted to spend more time at home with his wife and, if his productivity and team weren't being affected, just let him continue to do so for a while. He could have negotiated that Jake go into the office for one or two days a week until things settled down. Regardless of the end result, a little bit of empathy would have gone a long way toward keeping Jake happy in this situation.

帕布羅有許多方法可以以不同的方式處理這件事：他可以對傑克想花更多時間在家裡陪他的妻子表示一些理解，如果他的工作效率和團隊沒有受到影響，就讓他繼續保持一段時間。他可以協商讓傑克每週到辦公室工作一到兩天，直到事情解決為止。無論最終結果如何，在這種情況下，一點點同理心將大大有助於讓傑克保持快樂。

## Antipattern: Be Everyone's Friend 反模式：成為每個人的朋友

The first foray that most people have into leadership of any sort is when they become the manager or TL of a team of which they were formerly members. Many leads don't want to lose the friendships they've cultivated with their teams, so they will sometimes work extra hard to maintain friendships with their team members after becoming a team lead. This can be a recipe for disaster and for a lot of broken friendships. Don't confuse friendship with leading with a soft touch: when you hold power over someone's career, they might feel pressure to artificially reciprocate gestures of friendship.

大多數人第一次進入任何型別的領導層是當他們成為團隊的經理或 TL 時，他們以前是團隊的成員。許多領導不想失去他們與團隊培養起來的友誼，所以他們有時會在成為團隊領導後特別努力地維持與團隊成員的友誼。這可能會導致災難，也可能導致許多友誼破裂。不要把友誼和溫柔的領導混為一談：當你掌握了某人職業生涯的權力時，他們可能會感到壓力，需要收斂地回應友誼的姿態。

Remember that you can lead a team and build consensus without being a close friend of your team (or a monumental hard-ass). Likewise, you can be a tough leader without tossing your existing friendships to the wind. We've found that having lunch with your team can be an effective way to stay socially connected to them without making them uncomfortable—this gives you a chance to have informal conversations outside the normal work environment.

請記住，你可以領導一個團隊並建立共識，而不需要成為你團隊的親密朋友（或一個不折不扣的硬漢）。同樣，你也可以成為一個強硬的領導者，而不把你現有的友誼扔到九霄雲外。我們發現，與你的團隊共進午餐是一種有效的方式，既能與他們保持社交聯絡，又不會讓他們感到不舒服——這讓你有機會在正常工作環境之外進行非正式的對話。

Sometimes, it can be tricky to move into a management role over someone who has been a good friend and a peer. If the friend who is being managed is not self- managing and is not a hard worker, it can be stressful for everyone. We recommend that you avoid getting into this situation whenever possible, but if you can't, pay extra attention to your relationship with those folks.

有時，在一個曾經是好朋友和同齡人的人身上擔任管理職務可能會很棘手。如果被管理的朋友不具備自我管理的能力，也不是一個努力工作的人，那麼對每個人來說都會有壓力。我們建議你儘可能避免陷入這種情況，但如果你不能，就要特別注意你與這些人的關係。

## Antipattern: Compromise the Hiring Bar 反模式：妥協的招聘

Steve Jobs once said: "A people hire other A people; B people hire C people." It's incredibly easy to fall victim to this adage, and even more so when you're trying to hire quickly. A common approach I've seen outside of Google is that a team needs to hire 5 engineers, so it sifts through a pile of applications, interviews 40 or 50 people, and picks the best 5 candidates regardless of whether they meet the hiring bar.

史蒂夫·喬布斯曾經說過。"A 類人僱用其他 A 類人；B 類人僱用 C 類人。" 這句格言很容易讓你成為犧牲品，尤其是當你試圖快速僱傭的時候。我在 Google 之外看到的一個常見的方法是，一個團隊需要招聘 5 名工程師，所以它篩選了一堆申請，面試了 40 或 50 人，並挑選了最好的 5 名候選人，不管他們是否符合招聘標準。

This is one of the fastest ways to build a mediocre team.

這是建立一個平庸團隊的最快方式之一。

The cost of finding the appropriate person—whether by paying recruiters, paying for advertising, or pounding the pavement for references—pales in comparison to the cost of dealing with an employee who you never should have hired in the first place. This "cost" manifests itself in lost team productivity, team stress, time spent managing the employee up or out, and the paperwork and stress involved in firing the employee. That's assuming, of course, that you try to avoid the monumental cost of just leaving them on the team. If you're managing a team for which you don't have a say over hiring and you're unhappy with the hires being made for your team, you need to fight tooth and nail for higher-quality engineers. If you're still handed substandard engineers, maybe it's time to look for another job. Without the raw materials for a great team, you're doomed.

找到合適人選的成本——無論是透過支付招聘人員費用、支付廣告費用，還是為推薦人做鋪墊——與處理一個你一開始就不應該僱用的員工的成本相比，都顯得微不足道。這種 "成本 "體現在團隊生產效率的損失、團隊壓力、管理員工的時間以及解僱員工所涉及的文書工作和壓力上。當然，這是假設你試圖避免讓他們留在團隊中的巨大成本。如果你管理的團隊在招聘方面沒有發言權，而且你對你的團隊所招聘的人不滿意，你需要為更高素質的工程師拼命爭取。如果你仍然得到不合格的工程師，也許是時候尋找

另一份工作了。沒有優秀團隊的人才，你就註定要失敗。

## Antipattern: Treat Your Team Like Children 反模式：像對待孩子一樣對待你的團隊

The best way to show your team that you don't trust it is to treat team members like kids—people tend to act the way you treat them, so if you treat them like children or prisoners, don't be surprised when that's how they behave. You can manifest this behavior by micromanaging them or simply by being disrespectful of their abilities and giving them no opportunity to be responsible for their work. If it's permanently necessary to micromanage people because you don't trust them, you have a hiring failure on your hands. Well, it's a failure unless your goal was to build a team that you can spend the rest of your life babysitting. If you hire people worthy of trust and show these people you trust them, they'll usually rise to the occasion (sticking with the basic premise, as we mentioned earlier, that you've hired good people).

向你的團隊表明你不信任它的最好方法是把團隊成員當成小孩子——人們往往會按照你對待他們的方式行事，因此如果你像對待孩子或囚犯一樣對待他們，當他們的行為如此時，不要感到驚訝。你可以透過對他們進行微觀管理來體現這種行為，或者僅僅是不尊重他們的能力，不給他們機會對他們的工作負責。如果因為你不信任他們而長期需要對他們進行微觀管理，那麼你的招聘就失敗了。好吧，這是一個失敗，除非你的目標是建立一個你可以用餘生來照看的團隊。如果你僱用值得信任的人，並向這些人展示你對他們的信任，他們通常會迎難而上（堅持我們前面提到的基本前提，即你已經僱用了優秀的人）。

The results of this level of trust go all the way to more mundane things like office and computer supplies. As another example, Google provides employees with cabinets stocked with various and sundry office supplies (e.g., pens, notebooks, and other "legacy" implements of creation) that are free to take as employees need them. The IT department runs numerous "Tech Stops" that provide self-service areas that are like a mini electronics store. These contain lots of computer accessories and doodads (power supplies, cables, mice, USB drives, etc.) that would be easy to just grab and walk off with en masse, but because Google employees are being trusted to check these items out, they feel a responsibility to Do The Right Thing. Many people from typical corporations react in horror to hearing this, exclaiming that surely Google is hemorrhaging money due to people "stealing" these items. That's certainly possible, but what about the costs of having a workforce that behaves like children or that has to waste valuable time formally requesting cheap office supplies? Surely that's more expensive than the price of a few pens and USB cables.

這種信任程度的結果一直延伸到更平凡的事情，如辦公室和電腦用品。另一個例子是，Google 為員工提供了儲存有各種雜類辦公用品的櫃子（例如，筆、筆記本和其他 "傳統 "創作工具），員工可以根據需要自由取用。IT 部門經營著許多 "技術站"，提供自助服務區，就像一個小型電子產品商店。這些地方有很多電腦配件和小玩意（電源、電纜、滑鼠、隨身碟等），這些東西很容易就被拿走了，但因為 Google 員工被信任去獲取這些物品，他們覺得有責任去做正確的事情。許多來自典型企業的人聽到這個訊息後都會感到驚恐，他們感嘆說，他們驚呼肯定是因為有人"偷"這些東西，Google 肯定會損失慘重。這當然是可能的，但是擁有一支像小孩子一樣的員工隊伍，或者不得不浪費寶貴的時間正式申請廉價辦公用品的成本呢？這肯定比幾支筆和 USB 線的價格要貴。

# Positive Patterns 積極模式

Now that we've covered antipatterns, let's turn to positive patterns for successful leadership and management that we've learned from our experiences at Google, from watching other successful leaders and, most of all, from our own leadership mentors. These patterns are not only those that we've had great success implementing, but the patterns that we've always respected the most in the leaders who we follow.

現在我們已經介紹了反模式，讓我們來談談成功的領導和管理的積極模式，這些模式是我們從在 Google 的經驗中，從觀察其他成功的領導者中，最重要的是，從我們自己的領導導師那裡學到的。這些模式不僅是那些我們已經成功實施的模式，而且是我們一直以來最尊重的領導者的模式。

## Lose the Ego 丟掉自負

We talked about “losing the ego” a few chapters ago when we first examined humility, respect, and trust, but it’s especially important when you’re a team leader. This pattern is frequently misunderstood as encouraging people to be doormats and let others walk all over them, but that’s not the case at all. Of course, there’s a fine line between being humble and letting others take advantage of you, but humility is not the same as lacking confidence. You can still have self-confidence and opinions without being an egomaniac. Big personal egos are difficult to handle on any team, especially in the team’s leader. Instead, you should work to cultivate a strong collective team ego and identity.

我們在幾章前第一次研究謙卑、尊重和信任時談到了 "丟掉自負"，但當你是一個團隊領導時，這一點尤其重要。這種模式經常被誤解為鼓勵人們做受氣套件，讓別人踩在他們身上，但事實並非如此。當然，在謙虛和讓別人利用你之間有一條細微的界限，但謙虛不等於缺乏自信。你仍然可以有自信心和意見，而不是成為一個自大狂。在任何團隊中，特別是在團隊領導中，個人自大都是很難處理的。相反，你應該努力培養強大的集體自我和認同感。

Part of “losing the ego” is trust: you need to trust your team. That means respecting the abilities and prior accomplishments of the team members, even if they’re new to your team.

“丟失自負 "的一部分是信任：你需要信任你的團隊。這意味著尊重團隊成員的能力和先前的成就，即使他們是團隊的新成員。

If you’re not micromanaging your team, you can be pretty certain the folks working in the trenches know the details of their work better than you do. This means that although you might be the one driving the team to consensus and helping to set the direction, the nuts and bolts of how to accomplish your goals are best decided by the people who are putting the product together. This gives them not only a greater sense of ownership, but also a greater sense of accountability and responsibility for the success (or failure) of their product. If you have a good team and you let it set the bar for the quality and rate of its work, it will accomplish more than by you standing over team members with a carrot and a stick.

如果你沒有對你的團隊進行微觀管理，你可以非常肯定，那些在基層工作的人比你更瞭解他們工作的細節。這意味著，儘管你可能是推動團隊達成共識並幫助確定方向的人，但如何完成你的目標的具體細節最好是由正在製作產品的人決定。這不僅使他們有更大的主人翁意識，而且對他們產品的成功（或失敗）也有更大的責任感和使命感。如果你有一個好的團隊，並讓它為其工作的品質和速度設定標準，它將比你拿著胡蘿蔔和大棒站在團隊成員面前的成就更大。

Most people new to a leadership role feel an enormous responsibility to get everything right, to know everything, and to have all the answers. We can assure you that you will not get everything right, nor will you have all the answers, and if you act like you do, you’ll quickly lose the respect of your team. A lot of this comes down to having a basic sense of security in your role. Think back to when you were an individual contributor; you could smell insecurity a mile away. Try to appreciate inquiry: when someone questions a decision or statement you made, remember that this person is usually just trying to better understand you. If you encourage inquiry, you’re much more likely to get the kind of constructive criticism that will make you a better leader of a better team. Finding people who will give you good constructive criticism is incredibly difficult, and it’s even more difficult to get this kind of criticism from people who “work for you.” Think about the big picture of what you’re trying to accomplish as a team, and accept feedback and criticism openly; avoid the urge to be territorial.

大多數剛開始擔任領導角色的人都覺得自己肩負著巨大的責任，要做好每一件事，瞭解每一件事，並掌握所有答案。我們可以向你保證，你不會把所有事情都做對，也不會有所有的答案，如果你這樣做，你很快就會失去團隊的尊重。這很大程度上取決於你的角色是否具有基本的安全感。回想一下你還是個人貢獻者的時候，你在一英里外就能聞到不安全感。嘗試欣賞詢問：當有人質疑你的決定或宣告時，記住這個人通常只是想更好地瞭解你。如果你鼓勵詢問，你就更有可能得到那種建設性的批評，使你成為一個更好的團隊的領導者。找到會給你好的建設性批評的人是非常困難的，而從 "為你工作 "的人那裡得到這種批評就更難了。想一想你作為一個團隊所要完成的大局，坦然接受反饋和批評；避免地盤化的衝動。

The last part of losing the ego is a simple one, but many engineers would rather be boiled in oil than do it: apologize when you make a mistake. And we don't mean you should just sprinkle "I'm sorry" throughout your conversation like salt on popcorn—you need to sincerely mean it. You are absolutely going to make mistakes, and whether or not you admit it, your team is going to know you've made a mistake. Your team members will know regardless of whether they talk to you (and one thing is guaranteed: they *will* talk about it with one another). Apologizing doesn't cost money. People have enormous respect for leaders who apologize when they screw up, and contrary to popular belief, apologizing doesn't make you vulnerable. In fact, you'll usually gain respect from people when you apologize, because apologizing tells people that you are level headed, good at assessing situations, and—coming back to humility, respect, and trust—humble.

丟掉自負的最後一部分很簡單，但許多工程師寧願在油鍋裡也不願意這樣做：當你犯了錯誤時要道歉。我們並不是說你應該像在爆米花上撒鹽一樣在你的談話中撒下 "對不起"，你需要真誠地表達。你絕對會犯錯誤，無論你是否承認，你的團隊都會知道你犯了錯誤。無論你的團隊成員是否與你交談，他們都會知道（有一點是可以保證的：他們*會*彼此談論這個問題）。道歉不需要花錢。人們對那些在犯錯時道歉的領導人有著極大的尊重，與流行的觀點相反，道歉不會讓你變得脆弱。事實上，當你道歉時，你通常會得到人們的尊重，因為道歉告訴人們，你頭腦冷靜，善於評估情況，而且——回到謙遜、尊重和信任——謙虛。

## Be a Zen Master 管好自己

As an engineer, you've likely developed an excellent sense of skepticism and cynicism, but this can be a liability when you're trying to lead a team. This is not to say that you should be naively optimistic at every turn, but you would do well to be less vocally skeptical while still letting your team know you're aware of the intricacies and obstacles involved in your work. Mediating your reactions and maintaining your calm is more important as you lead more people, because your team will (both unconsciously and consciously) look to you for clues on how to act and react to whatever is going on around you.

作為一名工程師，你很可能已經養成了優秀的懷疑主義和憤世嫉俗的意識，但當你試圖領導一個團隊時，這可能是一種負擔。這並不是說你應該處處天真樂觀，但是你最好少說些懷疑的話，同時讓你的團隊知道你已經意識到了工作中的複雜性和障礙。當你領導更多的人時，調解你的反應和保持你的冷靜更加重要，因為你的團隊會（無意識地和有意識地）向你尋求線索，瞭解如何對你周圍發生的任何事情采取行動和作出反應。

A simple way to visualize this effect is to see your company's organization chart as a chain of gears, with the individual contributor as a tiny gear with just a few teeth all the way at one end, and each successive manager above them as another gear, ending with the CEO as the largest gear with many hundreds of teeth. This means that every time that individual's "manager gear" (with maybe a few dozen teeth) makes a single revolution, the "individual's gear" makes two or three revolutions. And the CEO can make a small movement and send the hapless employee, at the end of a chain of six or seven gears, spinning wildly! The farther you move up the chain, the faster you can set the gears below you spinning, whether or not you intend to.

將這種效應形象化的一個簡單方法是將你公司的組織結構圖看作是一個齒輪鏈，個人是一個很小的齒輪，只有幾個齒，而他們之上的每個繼任經理都是另一個齒輪，最後 CEO 是有數百顆牙的最大齒輪。這意味著個人的 "經理齒輪"（可能有幾十個齒）每轉一圈，"個人的齒輪 "就轉兩三圈。而執行長可以做一個小動作，讓處於六、七個齒輪鏈末端的無助的員工瘋狂地旋轉！你越是往上走，就越是如此。你在鏈條上走得越遠，你就能讓你下面的齒輪轉得越快，無論你是否打算這樣做。

Another way of thinking about this is the maxim that the leader is always on stage. This means that if you're in an overt leadership position, you are always being watched: not just when you run a meeting or give a talk, but even when you're just sitting at your desk answering emails. Your peers are watching you for subtle clues in your body language, your reactions to small talk, and your signals as you eat lunch. Do they read confidence or fear? As a leader, your job is to inspire, but inspiration is a 24/7 job. Your visible attitude about absolutely everything—no matter how trivial—is unconsciously noticed and spreads infectiously to your team.

另一種思考方式是 "領導者總是站在舞臺上" 的格言。這意味著，如果你處於公開的領導地位，你總是被監視著：不僅僅是當你主持會議或發表演講時，甚至當你只是坐在辦公桌前回覆電子郵件時。你的同僚在觀察你，從你的肢體語言、你對閒談的反應以及你吃午餐時的訊號中尋找微妙的線索。他們是讀出了自信還是恐懼？作為一個領導，你的工作是激勵，但激勵是一項全天候的工作。你對所有事情的明顯態度——無論多麼微不足道——都會不自覺地被注意到，並會傳染給你的團隊。

One of the early managers at Google, Bill Coughran, a VP of engineering, had truly mastered the ability to maintain calm at all times. No matter what blew up, no matter what crazy thing happened, no matter how big the firestorm, Bill would never panic. Most of the time he'd place one arm across his chest, rest his chin in his hand, and ask questions about the problem, usually to a completely panicked engineer. This had the effect of calming them and helping them to focus on solving the problem instead of running around like a chicken with its head cut off. Some of us used to joke that if someone came in and told Bill that 19 of the company's offices had been attacked by space aliens, Bill's response would be, "Any idea why they didn't make it an even 20?"

Google 的早期經理之一，工程部副部長比爾·考夫蘭，真正掌握了在任何時候都保持冷靜的能力。無論發生什麼事情，無論發生什麼瘋狂的事情，無論發生多大的風波，比爾都不會驚慌。大多數時候，他會把一隻手放在胸前，用手託著下巴，對問題進行提問，通常是向完全驚慌失措的工程師提問。這樣做的效果是讓他們平靜下來，幫助他們專注於解決問題，而不是像一隻被砍了頭的雞一樣到處亂跑。我們中的一些人曾經開玩笑說，如果有人進來告訴比爾，公司的 19 個辦公室被太空外星人襲擊了，比爾的反應會是："知道為什麼他們沒有襲擊 20 個？"。

This brings us to another Zen management trick: asking questions. When a team member asks you for advice, it's usually pretty exciting because you're finally getting the chance to fix something. That's exactly what you did for years before moving into a leadership position, so you usually go leaping into solution mode, but that is the last place you should be. The person asking for advice typically doesn't want *you* to solve their problem, but rather to help them solve it, and the easiest way to do this is to ask this person questions. This isn't to say that you should replace yourself with a Magic 8 Ball, which would be maddening and unhelpful. Instead, you can apply some humility, respect, and trust and try to help the person solve the problem on their own by trying to refine and explore the problem. This will usually lead the employee to the answer,[6] and it will be that person's answer, which leads back to the ownership and responsibility we went over earlier in this chapter. Whether or not you have the answer, using this technique will almost always leave the employee with the impression that you did. Tricky, eh? Socrates would be proud of you.

這給我們帶來了另一個管理自我技巧：問問題。當一個團隊成員向你徵求意見時，這通常是非常令人興奮的，因為你終於有機會解決一些問題了。這正是你在進入領導崗位之前多年所做的事情，所以你通常會立即進入解決方案模式，但這是你最不應該做

的。尋求建議的人通常不希望*你*解決他們的問題，而是希望你能幫助他們解決問題，而最簡單的方法就是向這個人提問。這並不是說你應該用 "魔力 8 球 "來代替你自己，那樣做會讓人發瘋，而且沒有幫助。相反，你可以運用一些謙遜、尊重和信任，透過嘗試完善和探索問題，嘗試幫助這個人自己解決這個問題。這通常會引導員工找到答案，而且這將是這個人的答案，這又回到了我們在本章前面所講的所有權和責任。無論你是否有答案，使用這種技巧幾乎總是會給員工留下你有答案的印象。很狡猾，是嗎？蘇格拉底會為你感到驕傲的。

> 6 另請參見“橡皮鴨除錯”

## Be a Catalyst 成為催化劑

In chemistry, a catalyst is something that accelerates a chemical reaction, but which itself is not consumed in the reaction. One of the ways in which catalysts (e.g., enzymes) work is to bring reactants into close proximity: instead of bouncing around randomly in a solution, the reactants are much more likely to favorably interact with one another when the catalyst helps bring them together. This is a role you'll often need to play as a leader, and there are a number of ways you can go about it.

在化學中，催化劑是加速化學反應的東西，但它本身在反應中不被消耗。催化劑（如酶）發揮作用的方式之一是使反應物接近：反應物不是在溶液中隨機地跳動，而是在催化劑的幫助下更有可能彼此有利地互動。這是你作為一個領導者經常需要扮演的角色，你可以透過多種方式來實現這一目標。

One of the most common things a team leader does is to build consensus. This might mean that you drive the process from start to finish, or you just give it a gentle push in the right direction to speed it up. Working to build team consensus is a leadership skill that is often used by unofficial leaders because it's one way you can lead without any actual authority. If you have the authority, you can direct and dictate direction, but that's less effective overall than building consensus.[7] If your team is looking to move quickly, sometimes it will voluntarily concede authority and direction to one or more team leads. Even though this might look like a dictatorship or oligarchy, when it's done voluntarily, it's a form of consensus.

團隊領導最常做的事情之一是建立共識。這可能意味著你從頭到尾推動這個過程，或者你只是在正確的方向上輕輕地推動它以加速它。努力建立團隊共識是一種領導技能，經常被非官方領導人使用，因為這是你可以在沒有任何實際權力的情況下進行領導的一種方式。如果你有權力，你可以指揮和發號施令，但這在整體上不如建立共識有效。如果你的團隊希望快速行動，有時會自願將權力和方向讓給一個或多個團隊領導。儘管這可能看起來像獨裁或寡頭政治，但當它是自願做的時候，它是一種共識的形式。

> 7 試圖達成 100%的共識也可能是有害的。你需要能夠決定繼續進行，即使不是每個人都在同一起跑線上，或者仍有一些不確定性。

## Remove Roadblocks 消除障礙

Sometimes, your team already has consensus about what you need to do, but it hit a roadblock and became stuck. This could be a technical or organizational roadblock, but jumping in to help the team get moving again is a common leadership technique. There are some roadblocks that, although virtually impossible for your team members to get past, will be easy for you to handle, and helping your team understand that you're glad (and able) to help out with these roadblocks is valuable.

有時，你的團隊已經對你需要做的事情達成了共識，但它遇到了障礙並陷入困境。這可能是一個技術上或組織上的障礙，但幫助團隊重新行動是一種常見的領導技巧。有一些障礙，雖然對你的團隊成員來說幾乎不可能逾越，但對你來說卻很容易處理，幫助你的團隊瞭解您樂於（並且能夠）幫助解決這些障礙是非常有價值的

One time, a team spent several weeks trying to work past an obstacle with Google's legal department. When the team finally reached its collective wits' end and went to its manager with the problem, the manager had it solved in less than two hours simply because he knew the right person to contact to discuss the matter. Another time, a team needed some server resources and just couldn't get them allocated. Fortunately, the team's manager was in communication with other teams across the company and managed to get the team exactly what it needed that very afternoon. Yet another time, one of the engineers was having trouble with an arcane bit of Java code. Although the team's manager was not a Java expert, she was able to connect the engineer to another engineer who knew exactly what the problem was. You don't need to know all the answers to help remove roadblocks, but it usually helps to know the people who do. In many cases, knowing the right person is more valuable than knowing the right answer.

有一次，一個團隊花了幾個星期的時間試圖克服 Google 法律部門的一個障礙。當該團隊最終一籌莫展，向其經理提出這個問題時，經理在不到兩個小時內就解決了這個問題，只因為他知道討論這個問題的正確關聯人。還有一次，一個團隊需要一些伺服器資源，卻無法將其分配。幸運的是，團隊經理與公司其他團隊進行了溝通，並設法在當天下午讓團隊完全滿足其需求。還有一次，一位工程師在處理一段神秘的 Java 程式碼時遇到了麻煩。雖然該團隊的經理不是 Java 專家，但她還是為該工程師聯絡到了另一位工程師，而這位工程師正是知道問題出在哪裡。你不需要知道所有的答案來幫助消除障礙，但是瞭解那些知道的人通常是有幫助的。在許多情況下，瞭解正確的人比知道正確的答案更有價值。

## Be a Teacher and a Mentor 成為一名教師和導師

One of the most difficult things to do as a TL is to watch a more junior team member spend 3 hours working on something that you know you can knock out in 20 minutes. Teaching people and giving them a chance to learn on their own can be incredibly difficult at first, but it's a vital component of effective leadership. This is especially important for new hires who, in addition to learning your team's technology and codebase, are learning your team's culture and the appropriate level of responsibility to assume. A good mentor must balance the trade-offs of a mentee's time learning versus their time contributing to their product as part of an effective effort to scale the team as it grows.

作為一名 TL，最困難的事情之一就是看著一名級別較低的團隊成員花 3 個小時做一些你知道可以在 20 分鐘內完成的事情。一開始，教人並給他們一個自學的機會可能非常困難，但這是有效領導的重要組成部分。這對於新員工尤其重要，他們除了學習團隊的技術和程式碼庫外，還學習團隊的文化和承擔的適當責任水平。一個好的導師必須權衡學員的學習時間與他們為產品貢獻的時間，作為有效努力的一部分，隨著團隊的發展擴大團隊規模。

Much like the role of manager, most people don't apply for the role of mentor—they usually become one when a leader is looking for someone to mentor a new team member. It doesn't take a lot of formal education or preparation to be a mentor. Primarily, you need three things: experience with your team's processes and systems, the ability to explain things to someone else, and the ability to gauge how much help your mentee needs. The last thing is probably the most important—giving your mentee enough information is what you're supposed to be doing, but if you overexplain things or ramble on endlessly, your mentee will probably tune you out rather than politely tell you they got it.

與經理的角色一樣，大多數人並不申請擔任導師的角色——他們通常在領導尋找指導新團隊成員的人時成為導師。要成為一名導師，不需要很多正式的教育或準備。主要來說，你需要三件事：對團隊的流程和系統的經驗，向別人解釋事情的能力，以及衡量被指導者需要多少幫助的能力。最後一點可能是最重要的——向被指導者提供足夠的資訊是你應該做的，但是如果你說得太多或者沒完沒了，被指導者可能會把你拒之門外，而不是禮貌地告訴你他們明白了。

## Set Clear Goals 制定明確的目標

This is one of those patterns that, as obvious as it sounds, is solidly ignored by an enormous number of leaders. If you're going to get your team moving rapidly in one direction, you need to make sure that every team member understands and agrees on what the direction is. Imagine your product is a big truck (and not a series of tubes). Each team member has in their hand a rope tied to the front of the truck, and as they work on the product, they'll pull the truck in their own direction. If your intention is to pull the truck (or product) northbound as quickly as possible, you can't have team members pulling every which way—you want them all pulling the truck north. If you're going to have clear goals, you need to set clear priorities and help your team decide how it should make trade-offs when the time comes.

這是其中的一種模式，儘管聽起來很明顯，但卻被大量領導者所忽視。如果你想讓你的團隊朝一個方向快速前進，你需要確保每個團隊成員都理解並同意這個方向。想象一下，你的產品是一輛大卡車（而不是一系列的管子）。每個團隊成員手裡都有一根綁在卡車前面的繩子，當他們在產品上工作時，他們會把卡車拉向自己的方向。如果你的目的是儘快將卡車（或產品）向北拉，你就不能讓團隊成員向各個方向拉，你要他們都把卡車拉到北邊。如果你想要有明確的目標，你需要設定明確的優先順序，並幫助你的團隊決定在時機成熟時應該如何權衡。

The easiest way to set a clear goal and get your team pulling the product in the same direction is to create a concise mission statement for the team. After you've helped the team define its direction and goals, you can step back and give it more autonomy, periodically checking in to make sure everyone is still on the right track. This not only frees up your time to handle other leadership tasks, it also drastically increases the efficiency of your team. Teams can (and do) succeed without clear goals, but they typically waste a great deal of energy as each team member pulls the product in a slightly different direction. This frustrates you, slows progress for the team, and forces you to use more and more of your own energy to correct the course.

指定一個明確的目標並讓你的團隊在同一個方向上拉動產品的最簡單的方法是為團隊建立一個簡潔的任務陳述。在你幫助團隊確定方向和目標後，你可以退後一步，給團隊更多的自主權，定期檢查，以確保每個人仍然在正確的軌道上。這不僅可以釋放你的時間來處理其他的領導任務，還可以大幅提高團隊的效率。如果沒有明確的目標，團隊可以（也確實）取得成功，但他們通常會浪費大量的精力，因為每個團隊成員將產品拉向稍微不同的方向。這會讓你感到沮喪，減慢團隊的進度，迫使你越來越多地使用自己的精力來糾正錯誤。

## Be Honest 以誠待人

This doesn't mean that we're assuming you are lying to your team, but it merits a mention because you'll inevitably find yourself in a position in which you can't tell your team something or, even worse, you need to tell everyone something they don't want to hear. One manager we know tells new team members, "I won't lie to you, but I will tell you when I can't tell you something or if I just don't know."

這並不意味著我們假設你在對你的團隊撒謊，但值得一提的是，你將不可避免地發現自己處於一種無法告訴團隊的境地，或者更糟糕的是，你需要告訴每個人他們不想聽的事情。我們認識的一位經理告訴新團隊成員，“我不會對你們撒謊，但當我不能告訴你們一些事情或者我只是不知道的什麼時候，可以告訴你們。”

If a team member approaches you about something you can't share, it's OK to just tell them you know the answer but are not at liberty to say anything. Even more common is when a team member asks you something you don't know the answer to: you can tell that person you don't know. This is another one of those things that seems blindingly obvious when you read it, but many people in a manager role feel that if they don't know the answer to something, it proves that they're weak or out of touch. In reality, the only thing it proves is that they're human.

如果一個團隊成員找你談一些你不能分享的事情，你可以告訴他們你知道答案，但不能隨意說話。更常見的是，當一個團隊成員問你一些你不知道的答案時：你可以告訴那個人你不知道。當你讀到它時，這是另一件看起來非常明顯的事情，但許多擔任經理角色的人覺得，如果他們不知道某事的答案，就證明他們軟弱或不合群。實際上，它唯一證明的是他們是人類別。

Giving hard feedback is…well, hard. The first time you need to tell one of your reports that they made a mistake or didn't do their job as well as expected can be incredibly stressful. Most management texts advise that you use the “compliment sandwich” to soften the blow when delivering hard feedback. A compliment sandwich looks something like this:

```
You're a solid member of the team and one of our smartest engineers. That being said, your code is convoluted and almost impossible for anyone else on the team to understand. But you've got great potential and a wicked cool T-shirt collection.
```

給予嚴厲的反饋是……嗯，很難。當你第一次需要告訴你的下屬他們犯了一個錯誤或沒有把工作做得像預期的那樣好時，可能會有難以置信的壓力。大多數管理學書籍建議你在提供硬反饋時使用 "讚美三明治 "來緩和打擊。讚美的三明治看起來像這樣：

```
你是團隊中一個可靠的成員，是我們最聰明的工程師之一。雖然如此，你的程式碼很複雜，團隊中的其他人幾乎不可能理解。但是，你有很大的潛力，而且有一件很酷的T恤衫收藏。
```

Sure, this softens the blow, but with this sort of beating around the bush, most people will walk out of this meeting thinking only, “Sweet! I've got cool T-shirts!” We *strongly* advise against using the compliment sandwich, not because we think you should be unnecessarily cruel or harsh, but because most people won't hear the critical message, which is that something needs to change. It's possible to employ respect here: be kind and empathetic when delivering constructive criticism without resorting to the compliment sandwich. In fact, kindness and empathy are critical if you want the recipient to hear the criticism and not immediately go on the defensive.

當然，這會減輕打擊，但在這種繞圈子的情況下，大多數人在離開會議時只會想，“太好了！我有很酷的 T 恤！”我們強烈建議不要使用讚美三明治，不是因為我們認為你應該不必要地殘忍或苛刻，而是因為大多數人不會聽到批評的資訊，也就是說有些東西需要改變。在這裡可以使用尊重：在發表建設性的批評時，不要求助於“讚美三明治”，而是要善意和同理心。事實上，如果你想讓接受者聽到批評而不是立即採取防禦措施，那麼善意和同理心是至關重要的。

Years ago, a colleague picked up a team member, Tim, from another manager who insisted that Tim was impossible to work with. He said that Tim never responded to feedback or criticism and instead just kept doing the same things he'd been told he shouldn't do. Our colleague sat in on a few of the manager's meetings with Tim to watch the interaction between the manager and Tim, and noticed that the manager made extensive use of the compliment sandwich so as not to hurt Tim's feelings. When they brought Tim onto their team, they sat down with him and very clearly explained that Tim needed to make some changes to work more effectively with the team:

> We're quite sure that you're not aware of this, but the way that you're interacting with the team is alienating and angering them, and if you want to be effective, you need to refine your communication skills, and we're committed to helping you do that.

幾年前，一位同事從另一位經理那裡接過了一個團隊成員蒂姆，這位經理堅持認為蒂姆是不能與之合作。他說，蒂姆從不回應反饋或批評，而只是不斷地做他被告知不應該做的事情。我們的同事旁聽了該經理與蒂姆的幾次會議，觀察該經理與蒂姆之間的互動，並注意到該經理為了不傷害蒂姆的感情，大量使用了讚美的三明治。當他們把蒂姆帶到他們的團隊時，他們與他坐下來，非常清楚地解釋蒂姆需要做出一些改變，以便更有效地與團隊合作：

> 我們很肯定你沒有意識到這一點，但你與團隊互動的方式正在疏遠和激怒他們，如果你想有效地工作，你需要改進你的溝通技巧，我們致力於幫助你做到這一點。

They didn't give Tim any compliments or candy-coat the issue, but just as important, they weren't mean—they just laid out the facts as they saw them based on Tim's performance with the previous team. Lo and behold, within a matter of weeks (and after a few more "refresher" meetings), Tim's performance improved dramatically. Tim just needed very clear feedback and direction.

在這個問題上，他們沒有給蒂姆任何讚揚或甜言蜜語，但同樣重要的是，他們並不意味著他們只是根據蒂姆在前一個團隊中的表現來陳述事實。你瞧，在幾周之內（以及在幾次"複習"會議之後），蒂姆的表現有了顯著的改善。蒂姆只需要非常清晰的反饋和指導。

When you're providing direct feedback or criticism, your delivery is key to making sure that your message is heard and not deflected. If you put the recipient on the defensive, they're not going to be thinking of how they can change, but rather how they can argue with you to show you that you're wrong. Our colleague Ben once managed an engineer who we'll call Dean. Dean had extremely strong opinions and would argue with the rest of the team about anything. It could be something as big as the team's mission or as small as the placement of a widget on a web page; Dean would argue with the same conviction and vehemence either way, and he refused to let anything slide. After months of this behavior, Ben met with Dean to explain to him that he was being too combative. Now, if Ben had just said, "Dean, stop being such a jerk," you can be pretty sure Dean would have disregarded it entirely. Ben thought hard about how he could get Dean to understand how his actions were adversely affecting the team, and he came up with the following metaphor:

> Every time a decision is made, it's like a train coming through town—when you jump in front of the train to stop it, you slow the train down and potentially annoy the engineer driving the train. A new train comes by every 15 minutes, and if you jump in front of every train, not only do you spend a lot of your time stopping trains, but eventually one of the engineers driving the train is going to get mad enough to run right over you. So, although it's OK to jump in front of some trains, pick and choose the ones you want to stop to make sure you're stopping only the trains that really matter.

當你提供直接的反饋或批評時，你的表達方式是確保你的資訊被聽到而不被偏離是關鍵。如果你讓接受者處於防守狀態，他們就不會考慮如何改變，而是會考慮如何與你爭辯以證明你錯了。我們的同事本曾經管理過一個工程師，我們稱之為迪安。迪安有非常強烈的意見，會和團隊的其他成員爭論任何事情。這件事可能大到團隊的任務，小到網頁上一個小部件的位置；無論如何，迪安都會以同樣的信念和激烈的態度進行爭論，而且他拒絕放過任何東西。這種行為持續了幾個月後，本與迪安見面，向他解釋說他太好鬥了。現在，如果本只是說："迪安，別再這麼混蛋了"，你可以很肯定迪安會完全不理會。本認真思考了如何讓迪安明白他的行為是如何對團隊產生不利影響的，他想出了下面這個比喻:

> 每次做出決定時，就像一列火車駛過小鎮—當你跳到火車前面去阻止它時，你就會使火車減速，並有可能惹惱駕駛火車的工程師。每15分鐘就會有一列新的火車經過，如果你在每一列火車前跳車，你不僅要花很多時間來阻止火車，而且最終駕駛火車的工程師眾人會生氣到直接從你身上碾過。因此，儘管跳到一些火車前面是可以的，但要挑選你想停的火車，以確保你只停真正重要的火車。

This anecdote not only injected a bit of humor into the situation, but also made it easier for Ben and Dean to discuss the effect that Dean's "train stopping" was having on the team in addition to the energy Dean was spending on it.

這段軼事不僅為情況注入了一點幽默感，而且使本和迪恩更容易討論迪恩的 "火車停擺 "除了耗費精力之外對團隊的影響。

## Track Happiness 追蹤幸福感

As a leader, one way you can make your team more productive (and less likely to leave) in the long term is to take some time to gauge their happiness. The best leaders we've worked with have all been amateur psychologists, looking in on their team members' welfare from time to time, making sure they get recognition for what they do, and trying to make certain they are happy with their work. One TLM we know makes a spreadsheet of all the grungy, thankless tasks that need to be done and makes certain these tasks are evenly spread across the team. Another TLM watches the hours his team is working and uses comp time and fun team outings to avoid burnout and exhaustion. Yet another starts one-on-one sessions with his team members by dealing with their technical issues as a way to break the ice, and then takes some time to make sure each engineer has everything they need to get their work done. After they've warmed up, he talks to the engineer for a bit about how they're enjoying the work and what they're looking forward to next.

作為一名領導者，從長遠來看，你可以讓你的團隊更有效率（也不太可能離開）的一種方法是花一些時間來衡量他們的幸福感。我們合作過的最好的領導都是業餘的心理學家，他們時常關注團隊成員的福利，確保他們的工作得到認可，並努力確保他們對工作感到滿意。我們認識的一位 TLM 將所有需要完成的煩人、吃力不討好的任務製成電子表格，並確保這些任務在團隊中平均分配。另一位 TLM 觀察他的團隊的工作時間，並利用補償時間和有趣的團隊外出活動來避免倦怠和疲憊。還有一個人開始與他的

團隊成員進行一對一的會談，處理他們的技術問題，以此來打破僵局，然後花一些時間來確保每個工程師擁有完成工作所需的一切。在他們熱身之後，他與工程師交談了一會兒，談論他們如何享受工作，以及他們接下來期待的事情。

A good simple way to track your team's happiness [8] is to ask the team member at the end of each one-on-one meeting, "What do you need?" This simple question is a great way to wrap up and make sure each team member has what they need to be productive and happy, although you might need to carefully probe a bit to get details. If you ask this every time you have a one-on-one, you'll find that eventually your team will remember this and sometimes even come to you with a laundry list of things it needs to make everyone's job better.

追蹤你的團隊幸福感的一個好的簡單方法是在每次一對一的會議結束時問團隊成員："你需要什麼？" 這個簡單的問題是一個很好的總結方式，確保每個團隊成員都有他們需要的東西，以提高工作效率和幸福感，儘管你可能需要仔細探究一下以獲得細節。如果你在每次一對一會談時都這樣問，你會發現最終你的團隊會記住這一點，有時甚至會帶著一長串需要的東西來找你，以使每個人的工作變得更好。

> 8 Google 還開展了一項名為 "Googlegeist "的年度員工調查，從多個方面對員工的幸福感進行評價。這提供了良好的反饋，但並不是我們所說的 "簡單"。

## The Unexpected Question 意想不到的問題

Shortly after I started at Google, I had my first meeting with then-CEO Eric Schmidt, and at the end Eric asked me, "Is there anything you need?" I had prepared a million defensive responses to difficult questions or challenges but was completely unprepared for this. So I sat there, dumbstruck and staring. You can be sure I had something ready the next time I was asked that question!

在我進入 Google 後不久，我與當時的執行長埃裡克·施密特（Eric Schmidt）進行了第一次會面，最後埃裡克問我："你需要什麼嗎？"我準備了一百萬份針對困難問題或挑戰的防禦性回覆，但對此完全沒有準備。於是我坐在那裡，呆呆地望著。下次再被問到這個問題時，我已經準備好了東西！'

It can also be worthwhile as a leader to pay some attention to your team's happiness outside the office. Our colleague Mekka starts his one-on-ones by asking his reports to rate their happiness on a scale of 1 to 10, and oftentimes his reports will use this as a way to discuss happiness in *and* outside of the office. Be wary of assuming that people have no life outside of work—having unrealistic expectations about the amount of time people can put into their work will cause people to lose respect for you, or worse, to burn out. We're not advocating that you pry into your team members' personal lives, but being sensitive to personal situations that your team members are going through can give you a lot of insight as to why they might be more or less productive at any given time. Giving a little extra slack to a team member who is currently having a tough time at home can make them a lot more willing to put in longer hours when your team has a tight deadline to hit later.

作為一個領導者，關注一下你的團隊在辦公室以外的幸福感也是值得的。我們的同事梅卡在他的一對一談話中，首先要求他的報告在 1 到 10 的範圍內給他們的幸福感打分，而他的報告往往會以此作為一種方式來討論辦公室內外的幸福。要警惕假設人們沒有工作以外的生活——對人們能夠投入工作的時間有不切實際的期望，會導致人們失去對你的尊重，或者更糟糕的是，會讓人倦怠。我們並不提倡你窺探團隊成員的私人生活，但對團隊成員正在經歷的個人情況保持敏感，可以讓你深入瞭解為什麼他們在任何特定時間可能會更有或更沒有生產效率。給目前在家裡過得很艱難的團隊成員一點額外的寬容，可以使他們在你的團隊以後有

一個緊迫的截止日期時更願意投入更多的時間。

A big part of tracking your team members' happiness is tracking their careers. If you ask a team member where they see their career in five years, most of the time you'll get a shrug and a blank look. When put on the spot, most people won't say much about this, but there are usually a few things that everyone would like to do in the next five years: be promoted, learn something new, launch something important, and work with smart people. Regardless of whether they verbalize this, most people are thinking about it. If you're going to be an effective leader, you should be thinking about how you can help make all those things happen and let your team know you're thinking about this. The most important part of this is to take these implicit goals and make them explicit so that when you're giving career advice you have a real set of metrics with which to evaluate situations and opportunities.

追蹤你的團隊成員的幸福感的一個重要部分是追蹤他們的職業生涯。如果你問一個團隊成員他們對五年後的職業生涯的看法，大多數時候你會得到一個聳肩和一個茫然的眼神。當被問及這個問題時，大多數人都不會多說什麼，但通常有幾件事是每個人在未來五年都想做的：晉升、學習新東西、推出重要的東西、與聰明人一起工作。不管他們是否口頭上這麼說，大多數人都在考慮這個問題。如果你要成為一個有效的領導者，你應該考慮如何幫助實現所有這些事情，並讓你的團隊知道你在考慮這個問題。其中最重要的部分是將這些隱含的目標明確化，這樣當你提供職業建議時，你就有了一套真正的衡量標準，用來評估形勢和機會。

Tracking happiness comes down to not just monitoring careers, but also giving your team members opportunities to improve themselves, be recognized for the work they do, and have a little fun along the way.

追蹤幸福感歸根結底不僅僅是監測職業，還要給你的團隊成員提供機會來提高自己，使他們的工作得到認可，並在此過程中獲得一些樂趣。

# Other Tips and Tricks 其他提示和竅門

Following are other miscellaneous tips and tricks that we at Google recommend when you're in a leadership position:

- *Delegate, but get your hands dirty*
  When moving from an individual contributor role to a leadership role, achieving a balance is one of the most difficult things to do. Initially, you're inclined to do all of the work yourself, and after being in a leadership role for a long time, it's easy to get into the habit of doing none of the work yourself. If you're new to a leadership role, you probably need to work hard to delegate work to other engineers on your team, even if it will take them a lot longer than you to accomplish that work. Not only is this one way for you to maintain your sanity, but also it's how the rest of your team will learn. If you've been leading teams for a while or if you pick up a new team, one of the easiest ways to gain the team's respect and get up to speed on what they're doing is to get your hands dirty—usually by taking on a grungy task that no one else wants to do. You can have a resume and a list of achievements a mile long, but nothing lets a team know how skillful and dedicated (and humble) you are like jumping in and actually doing some hard work.
- *Seek to replace yourself*
  Unless you want to keep doing the exact same job for the rest of your career, seek to replace yourself. This starts, as we mentioned earlier, with the hiring process: if you want a member of your team to replace you, you need to hire people capable of replacing you, which we usually sum up by saying that you need to "hire people smarter than you." After you have team members capable of doing your job, you need to give them opportunities to take on more responsibilities or occasionally lead the team. If you do this, you'll quickly see who has the most aptitude to lead as well as who wants to

lead the team. Remember that some people prefer to just be high-performing individual contributors, and that's OK. We've always been amazed at companies that take their best engineers and—against their wishes—throw these engineers into management roles. This usually subtracts a great engineer from your team and adds a subpar manager.

- *Know when to make waves*
  You will (inevitably and frequently) have difficult situations crop up in which every cell in your body is screaming at you to do nothing about it. It might be the engineer on your team whose technical chops aren't up to par. It might be the person who jumps in front of every train. It might be the unmotivated employee who is working 30 hours a week. "Just wait a bit and it will get better," you'll tell yourself. "It will work itself out," you'll rationalize. Don't fall into this trap—these are the situations for which you need to make the biggest waves and you need to make them now. Rarely will these problems work themselves out, and the longer you wait to address them, the more they'll adversely affect the rest of the team and the more they'll keep you up at night thinking about them. By waiting, you're only delaying the inevitable and causing untold damage in the process. So act, and act quickly.
- *Shield your team from chaos*
  When you step into a leadership role, the first thing you'll usually discover is that outside your team is a world of chaos and uncertainty (or even insanity) that you never saw when you were an individual contributor. When I first became a manager back in the 1990s (before going back to being an individual contributor), I was taken aback by the sheer volume of uncertainty and organizational chaos that was happening in my company. I asked another manager what had caused this sudden rockiness in the otherwise calm company, and the other manager laughed hysterically at my naivete: the chaos had always been present, but my previous manager had shielded me and the rest of my team from it.
- *Give your team air cover*
  Whereas it's important that you keep your team informed about what's going on "above" them in the company, it's just as important that you defend them from a lot of the uncertainty and frivolous demands that can be imposed upon you from outside your team. Share as much information as you can with your team, but don't distract them with organizational craziness that is extremely unlikely to ever actually affect them.
- *Let your team know when they're doing well*
  Many new team leads can get so caught up in dealing with the shortcomings of their team members that they neglect to provide positive feedback often enough. Just as you let someone know when they screw up, be sure to let them know when they do well, and be sure to let them (and the rest of the team) know when they knock one out of the park.

下面是 Google 在你擔任領導職務時推薦的其他提示和竅門：

- *委託，但要弄髒自己的手*
  當從個人貢獻者的角色轉變為領導角色時，實現平衡是最難做到的事情之一。起初，你會傾向於自己做所有的工作，而在領導崗位上呆久了，很容易養成自己不做任何工作的習慣。如果你剛開始擔任領導職務，你可能需要努力工作，把工作委託給團隊中的其他工程師，即使他們完成這項工作所需的時間比你長很多。這不僅是你保持理智的一種方式，而且也是你的團隊其他成員學習的方式。如果你已經領導了一段時間的團隊，或者你接了一個新的團隊，獲得團隊的尊重和了解他們的工作的最簡單的方法之一就是弄髒你的手——通常是承擔一個別人不願意做的骯髒的任務。你可以有一份簡歷和一份一英里長的成就清單，但沒有任何東西能讓團隊知道你有多熟練、有多謙遜（和謙遜），你喜歡跳進去做一些艱苦的工作。
- *尋求繼任者*
  除非你想在餘下的職業生涯中一直做著完全相同的工作，否則要設法尋找繼任者。正如我們前面提到的，這從招聘過程開始：如果你想讓你的團隊成員取代你，你需要僱傭有能力取代你的人，我們通常總結說，你需要 "僱傭比你更聰明的人"。在你擁有能夠勝任工作的團隊成員之後，你需要給他們機會承擔更多的責任或偶爾領導團隊。如果你這樣做，你會很快看

到誰最有領導才能，以及誰想領導團隊。請記住，有些人更願意只做高績效的個人，這也是可以的。我們一直對一些公司感到驚訝，這些公司把他們最優秀的工程師，違揹他們的意願，把這些工程師扔到管理崗位上。這通常會從你的團隊中減少一名優秀的工程師，而增加一名不合格的經理。

- *知道什麼時候該掀起風波*
  你會（不可避免且經常地）遇到一些困難的情況，在這些情況下，你身體裡的每一個細胞都在對你大喊大叫，要求你什麼都不要做。這可能是你團隊中的工程師，他的技術能力達不到要求。它可能是那個在每輛火車前跳來跳去的人。它可能是每週工作 30 小時的無心的員工。"只要等一等，就會好起來的，"你會告訴自己。"它會自己解決的，"你會合理地解釋。不要落入這個陷阱——這些是你需要掀起最大波瀾的情況，你需要現在就掀起。這些問題很少會自己解決，你等待解決的時間越長，它們對團隊其他成員的不利影響就越大，它們會讓你徹夜思考。透過等待，你只是拖延了不可避免的事情，並在這個過程中造成難以言喻的損失。因此，要採取行動，而且要迅速行動。
- *遮蔽團隊免受混亂影響*
  當你步入領導崗位時，你通常會發現，在你的團隊之外是一個混亂和不確定（甚至是瘋狂）的世界，而你在做個人貢獻者時從未見過。當我在 20 世紀 90 年代第一次成為一名經理時（在回到個人貢獻者之前），我對公司裡發生的大量不確定性和組織混亂感到吃驚。我問另一位經理，是什麼原因導致原本平靜的公司突然出現這種動盪，另一位經理歇斯底里地笑我太天真了：混亂一直存在，但我以前的經理把我和我的團隊其他成員都擋在外面。
- *給你的團隊提供空中掩護*
  儘管讓你的團隊瞭解公司 "上面 "發生的事情很重要，但同樣重要的是，你要保護他們不受很多不確定因素和輕率要求的影響，這些要求可能來自你的團隊之外。儘可能多地與你的團隊分享資訊，但不要用那些極不可能真正影響到他們的組織的瘋狂行為來分散他們的注意力。
- *讓你的團隊知道他們什麼時候做得好*
  許多新的團隊領導可能會陷入處理團隊成員的缺點中，以至於他們忽略了經常提供積極的反饋。就像你讓某人知道他們搞砸了一樣，一定要讓他們知道他們做得很好，而且一定要讓他們（和團隊其他成員）知道他們在球場裡踢出了一個好成績。

Lastly, here's something the best leaders know and use often when they have adventurous team members who want to try new things:

*It's easy to say "yes" to something that's easy to undo*

If you have a team member who wants to take a day or two to try using a new tool or library [9] that could speed up your product (and you're not on a tight deadline), it's easy to say, "Sure, give it a shot." If, on the other hand, they want to do something like launch a product that you're going to have to support for the next 10 years, you'll likely want to give it a bit more thought. Really good leaders have a good sense for when something can be undone, but more things are undoable than you think (and this applies to both technical and nontechnical decisions).

最後，這裡有一些最好的領導者知道的東西，當他們有想嘗試新事物的富有冒險精神的團隊成員時，他們經常使用：

*很容易對容易撤銷的事情說 "是 "*
如果你有一個團隊成員想花一兩天時間嘗試使用一個新的工具或庫，可以加速你的產品（而且你沒有緊迫的期限），你很容易說："當然，給它一個機會。" 另一方面，如果他們想做一些事情，比如推出一個你必須在未來 10 年內支援的產品，你可能會想多考慮一下。真正好的領導對什麼時候可以撤銷的事情有很好的感覺，但更多的事情是可以撤銷的，而不是你想象的那樣（這適用於技術和非技術的決定）。

> 9 要想更好地瞭解技術變革是如何 "不可逆轉 "的，見第 22 章。

## People Are Like Plants 人如植物

My wife is the youngest of six children, and her mother was faced with the difficult task of figuring out how to raise six very different children, each of whom needed different things. I asked my mother-in-law how she managed this (see what I did there?), and she responded that kids are like plants: some are like cacti and need little water but lots of sunshine, others are like African violets and need diffuse light and moist soil, and still others are like tomatoes and will truly excel if you give them a little fertilizer. If you have six kids and give each one the same amount of water, light, and fertilizer, they'll all get equal treatment, but the odds are good that *none* of them will get what they actually need.

我的妻子是六個孩子中最小的一個，她的母親面臨著一項艱鉅的任務，即如何撫養六個完全不同的孩子，每個孩子都需要不同的東西。我問我的岳母她是如何做到的（看到我在那裡做了什麼了嗎？），她回答說，孩子就像植物：有些像仙人掌，需要很少的水，但需要大量的陽光；有些像非洲紫羅蘭，需要漫射的光和溼潤的土壤；還有一些像西紅柿，如果你給他們一點肥料，他們會非常出色。如果你有六個孩子，並且給每個孩子相同數量的水、光和肥料，他們都會得到同等的待遇，但很有可能他們都得不到他們真正需要的東西。

And so your team members are also like plants: some need more light, and some need more water (and some need more… fertilizer). It's your job as their leader to determine who needs what and then give it to them—except instead of light, water, and fertilizer, your team needs varying amounts of motivation and direction.

因此，你的團隊成員也像植物一樣：有些需要更多的光，有些需要更多的水（有些需要更多的.....肥料）。作為他們的領導，你的工作是確定誰需要什麼，然後給他們——只是你的團隊需要的不是光、水和肥料，而是不同程度的動力和方向。

To get all of your team members what they need, you need to motivate the ones who are in a rut and provide stronger direction to those who are distracted or uncertain of what to do. Of course, there are those who are "adrift" and need both motivation and direction. So, with this combination of motivation and direction, you can make your team happy and productive. And you don't want to give them too much of either—because if they don't need motivation or direction and you try giving it to them, you're just going to annoy them.

為了讓所有團隊成員都得到他們所需要的，你需要激勵那些墨守成規的人，併為那些分心或不確定該做什麼的人提供更有力的指導。當然，有些人是"漂泊"的，需要動力和方向。因此，透過這種動機和方向的結合，你可以讓你的團隊快樂並富有成效。你也不想給他們太多，因為如果他們不需要動力或方向，而你試圖給他們動力或方向，你只會惹惱他們。

Giving direction is fairly straightforward—it requires a basic understanding of what needs to be done, some simple organizational skills, and enough coordination to break it down into manageable tasks. With these tools in hand, you can provide sufficient guidance for an engineer in need of directional help. Motivation, however, is a bit more sophisticated and merits some explanation.

給出方向是相當簡單的——它需要對需要做什麼有一個基本的瞭解，一些簡單的組織技能，以及足夠的協調來將其分解為可管理的任務。有了這些工具，你可以為需要定向幫助的工程師提供足夠的指導。然而，動機有點複雜，值得解釋一下。

## Intrinsic Versus Extrinsic Motivation 內在動機與外在動機的關係

There are two types of motivation: *extrinsic*, which originates from outside forces (such as monetary compensation), and *intrinsic*, which comes from within. In his book *Drive*,[10] Dan Pink explains that the way to make people the happiest and most productive isn't to motivate them extrinsically (e.g., throw piles of cash at them); rather, you need to work to increase their intrinsic motivation. Dan claims you can increase intrinsic motivation by giving people three things: autonomy, mastery, and purpose.[11]

有兩種型別的動機：*外部動機*，來源於外部力量（如金錢補償），和*內部動機*，來源於內部。丹-平克在他的書《驅動》中解釋說，讓人們成為最快樂、最有效率的人的方法不是外在地激勵他們（例如，向他們扔大量現金）；相反，你需要努力提高他們的內在動機。丹聲稱，你可以透過給人們三樣東西來增加內在動機：自主性、掌控力和目標。

A person has autonomy when they have the ability to act on their own without someone micromanaging them.[12] With autonomous employees (and Google strives to hire mostly autonomous engineers), you might give them the general direction in which they need to take the product but leave it up to them to decide how to get there. This helps with motivation not only because they have a closer relationship with the product (and likely know better than you how to build it), but also because it gives them a much greater sense of ownership of the product. The bigger their stake is in the success of the product, the greater their interest is in seeing it succeed.

當一個人能夠獨立行動而不受任何人的微觀管理時，他就擁有了自主權。有了自主員工（Google 努力僱傭的大多是自主工程師），你可能會給他們提供他們需要的產品的大方向，但讓他們自己決定如何去做。這有助於激勵，不僅因為他們與產品的關係更密切（而且可能比你更瞭解如何建構產品），而且還因為這讓他們對產品有更大的主人翁意識。他們對產品的成功所持的股份越大，他們對看到產品成功的興趣就越大。

Mastery in its basest form simply means that you need to give someone the opportunity to improve existing skills and learn new ones. Giving ample opportunities for mastery not only helps to motivate people, it also makes them better over time, which makes for stronger teams.[13] An employee's skills are like the blade of a knife: you can spend tens of thousands of dollars to find people with the sharpest skills for your team, but if you use that knife for years without sharpening it, you will wind up with a dull knife that is inefficient, and in some cases useless. Google gives ample opportunities for engineers to learn new things and master their craft so as to keep them sharp, efficient, and effective.

掌握最基本的方式只意味著你需要給某人機會來提高現有技能並學習新技能。提供充分的掌握技能的機會不僅有助於激勵員工，而且隨著時間的推移，他們也會變得更好，從而形成更強大的團隊。員工的技能就像刀鋒：你可以花數萬美元為團隊找到最有技能的人，但是，如果你使用這把刀多年而沒有磨快它，你會得到一把鈍刀，這是低效的，在某些情況下是無用的。Google 為工程師提供了大量學習新事物和掌握技能的機會，從而使他們保持敏銳、高效和有效。

Of course, all the autonomy and mastery in the world isn't going to help motivate someone if they're doing work for no reason at all, which is why you need to give their work purpose. Many people work on products that have great significance, but they're kept at arm's length from the positive effects their products might have on their company, their customers, or even the world. Even for cases in which the product might have a much smaller impact, you can motivate your team by seeking the reason for their efforts and making this reason clear to them. If you can help them to see this purpose in their work, you'll see a tremendous increase in their motivation and productivity.14 One manager we know keeps a close eye on the email feedback that Google gets for its product (one of the "smaller-impact" products), and whenever she sees a message from a customer

talking about how the company's product has helped them personally or helped their business, she immediately forwards it to the engineering team. This not only motivates the team, but also frequently inspires team members to think about ways in which they can make their product even better.

當然，如果某人無緣無故地工作，那麼世界上所有的自主性和掌握性都無助於激勵他們，這就是為什麼你需要給他們工作的目的。許多人從事具有重大意義的產品，但他們與產品可能對公司、客戶甚至世界產生的積極影響保持一定距離。即使在產品的影響可能小得多的情況下，你也可以透過尋找他們努力的原因並向他們說明原因來激勵團隊。如果你能幫助他們在工作中看到這一目標，你會看到他們的積極性和生產效率有了巨大的提高。我們認識的一位經理密切關注 Google 為其產品（一種"影響較小"的產品）收到的電子郵件反饋，每當她看到客戶在談論公司的產品如何幫助他們個人或幫助他們的業務時，她會立即將其轉發給工程團隊。這不僅激勵了團隊，也經常激勵團隊成員思考如何讓他們的產品變得更好。

> 10 看丹關於這個話題的精彩 TED 演講。
>
> 11 這是假設相關人員的薪酬足夠高，收入不是壓力來源。
>
> 12 這假設您的團隊中有不需要微觀管理的人員。
>
> 13 當然，這也意味著他們是更有價值、更有市場價值的員工，因此，如果他們不喜歡自己的工作，他們會更容易接你離開。請參見第 99 頁"追蹤幸福感"中的模式。
>
> 14 Adam M.Grant，"任務重要性的重要性：工作績效影響、關係機制和邊界條件"，《應用心理學雜誌》，第 93 期，第 1 期（2018 年），http://bit.ly/task_significance.

## Conclusion 總結

Leading a team is a different task than that of being a software engineer. As a result, good software engineers do not always make good managers, and that's OK— effective organizations allow productive career paths for both individual contributors and people managers. Although Google has found that software engineering experience itself is invaluable for managers, the most important skills an effective manager brings to the table are social ones. Good managers enable their engineering teams by helping them work well, keeping them focused on proper goals, and insulating them from problems outside the group, all while following the three pillars of humility, trust, and respect.

領導團隊與作為軟體工程師是不同的任務。因此，好的軟體工程師並不總是能成為好的管理者，這也沒關係——高效的組織為個人貢獻者和人員管理者提供了富有成效的職業道路。儘管 Google 發現軟體工程經驗本身對管理者來說是無價的，但一個有效的管理者所帶來的最重要的技能是社交技能。優秀的管理者幫助他們的工程團隊做好工作，讓他們專注於正確的目標，讓他們遠離團隊之外的問題，同時遵循謙遜、信任和尊重這三大支柱。

# TL;DRs 內容提要

- Don’t “manage” in the traditional sense; focus on leadership, influence, and serving your team.
- Delegate where possible; don’t DIY (Do It Yourself).
- Pay particular attention to the focus, direction, and velocity of your team.
- 不要進行傳統意義上的 "管理"；重點是領導力、影響力和為你的團隊服務。
- 儘可能地授權；不要 DIY（自己動手）。
- 特別注意你的團隊的重點、方向和效率。

---

1. Another difference that takes getting used to is that the things we do as managers typically pay off over a longer timeline. ↩

2. Yet another reason companies shouldn’t force people into management as part of a career path: if an engineer is able to write reams of great code and has no desire at all to manage people or lead a team, by forcing them into a management or TL role, you’re losing a great engineer and gaining a crappy manager. This is not only a bad idea, but it’s actively harmful. ↩

3. For more fascinating information on optimizing the movements of factory workers, read up on Scientific Management or Taylorism, especially its effects on worker morale. ↩

4. If you have kids, the odds are good that you can remember with startling clarity the first time you said something to your child that made you stop and exclaim (perhaps even aloud), “Holy crap, I’ve become my mother.” ↩

5. Public criticism of an individual is not only ineffective (it puts people on the defense), but rarely necessary, and most often is just mean or cruel. You can be sure the rest of the team already knows when an individual has failed, so there’s no need to rub it in. ↩

6. See also “Rubber duck debugging.” ↩

7. Attempting to achieve 100% consensus can also be harmful. You need to be able to decide to proceed even if not everyone is on the same page or there is still some uncertainty. ↩

8. Google also runs an annual employee survey called “Googlegeist” that rates employee happiness across many dimensions. This provides good feedback but isn’t what we would call “simple.” ↩

9. To gain a better understanding of just how “undoable” technical changes can be, see Chapter 22. ↩

10. See Dan’s fantastic TED talk on this subject. ↩

11. This assumes that the people in question are being paid well enough that income is not a source of stress. ↩

12. This assumes that you have people on your team who don’t need micromanagement. ↩

13. Of course, it also means they’re more valuable and marketable employees, so it’s easier for them to pick up and leave you if they’re not enjoying their work. See the pattern in “Track Happiness” on page 99. ↩