《代码英雄》第三季（5）： 基础设施的影响

代码英雄讲述了开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。

什么是《代码英雄》

Command Line Heroes

代 码 英 雄是世界领先的企业开源软件解决方案供应商红帽（Red Hat）精心制作的原创音频播客，讲述开发人员、程序员、黑客、极客和开源反叛者如何彻底改变技术前景的真实史诗。该音频博客邀请到了谷歌、NASA 等重量级企业的众多技术大牛共同讲述开源、操作系统、容器、DevOps、混合云等发展过程中的动人故事。

本文是《代码英雄》系列播客《代码英雄》第三季（5）：基础设施的影响的音频脚本。

导语：用在 IT 基础设施中的语言是没有有效期的。COBOL 已经存在了 60 年 —— 而且不会很快消失。我们为大型机维护了数十亿行经典代码。但我们也在用 Go 等语言为云构建新的基础设施。

COBOL 是计算机的一次巨大飞跃，让各行各业变得更加高效。Chris Short 介绍了学习 COBOL 是如何被视为安全的长期投注的。60 年后的今天，还有数十亿行并不容易被替换掉的 COBOL 代码 —— 懂这门语言的专家也很少。Ritika Trikha 解释说，有些事情必须改变。要么更多的人必须学习 COBOL，要么依赖 COBOL 的行业必须更新他们的代码库。这两个选择都很困难。但未来并不是用 COBOL 编写的。今天的 IT 基础架构是在云端构建的，其中很多是用 Go 编写的。Carmen Hernández Andoh 分享了 Go 的设计者是如何想要设计一种更适合云计算的语言。

Kelsey Hightower 指出，语言通常都是超专注于一种任务。但它们正变得越来越开放和灵活。

00:00:00 - Saron Yitbarek：

1904 年，纽约市地铁首次开始运营时，它被惊叹为现代的一个奇迹。但是……当今天的通勤者仍依赖一个多世纪前设计的基础设施时，会发生什么？列车挤满了人，而且常常晚点。纽约每年有 20 亿人次地铁出行，再也没有人为此感到惊叹了。我们还在依赖昨日的过时基础设施，我们必须找到新的好办法，让它发挥作用。

00:00:44：

过去，基础设施项目通常是些可见的大而具体的事物，例如地铁。而且由于这种物理可见性，它们损坏时也显而易见。高速公路开裂、电线杆倒下，我们很清楚这些东西何时需要维修。为了使我们的生活与老化的基础设施保持协调，大量的工作是必不可少的。

00:01:12：

但是事物并不总是那么一是一、二是二。如今，我们还拥有 IT 基础设施，在偏僻地区嗡嗡作响的服务器农场，跨越海洋的光纤电缆，还有软件基础设施。而像遗留的操作系统或没人敢替换的 shell 脚本，这些 IT 基础设施变得陈旧和破旧时，我们是无法看出的。但是，造就了今日发展的基础设施却正在老化，就像旧的地铁轨道一样。这可能会扰乱我们的现代生活。如今命令行英雄们正努力确保我们不会被过去束缚，因此出现了大量新的挑战。

00:02:02：

这是我们第三季进入编程语言世界探索的第 5 期。我们将研究两种编程语言，它们与最初设计的目标基础设施密切相关。COBOL 是一种大型机的原生语言，而 Go 是云计算的原生语言。它们都深受其起源的影响。理解这一点可能会让明天的开发者不至于像纽约人一样被塞进宾夕法尼亚州的车站。

00:02:33：

我是 Saron Yitbarek，这里是红帽的原创播客，《命令行英雄》的第三季。

00:02:43 - 格蕾丝·赫柏：

我们面前有很多事情需要去做，但是我们首先需要大量相关的且易于访问的信息。我们才刚刚开始。

00:02:53 - Saron Yitbarek：

海军上将格蕾丝·赫柏
在 20 世纪 40、50 年代率先开发了高级编程语言。而她之所以能够实现这种巨大的飞跃，是因为当时的基础设施，大型计算机。

00:03:08 - Chris Short：

嗨，我叫 Chris Short。

00:03:10 - Saron Yitbarek：

Chris 是红帽的首席产品营销经理，而且他也是一位历史爱好者。

00:03:17 - Chris Short：

上世纪 40 年代的赫柏上将创造了 FLOW-MATIC，这在当时是革命性的，她被广泛认为是 COBOL 的祖母。因此，她能够坐在那里说：“嘿，只需将其放在大型机上”，或“嘿，只需将其存储在大型机上”即可。

00:03:31 - Saron Yitbarek：

这是一个重大的游戏规则改变。突然，你有了这种机器无关的语言，即 COBOL，它是大型机环境特有的。可能性开始逐步打开。

00:03:42 - Chris Short：

大型机和 COBOL 真正使得每个组织能够说，它们不再需要充满了带着铅笔、纸、计算器和滑尺的人的办公室，他们可能只需要半个办公

室来安装大型机。然后，他们可以雇人用 COBOL 来编写一些应用程序来完成整个财务团队做的所有的数学、逻辑运算以及账目。因此，你需要的财务团队人数少了，仅仅是因为更多的输入可以被数字化，而不是全手动操作。

00:04:17 - Saron Yitbarek：

如果你是那些新来的 COBOL 程序员之一，你会觉得你有了一份终身的工作。因为你的工作所基于的基础设施 —— 所有那些大型机 —— 始终会在那里。

00:04:30 - Chris Short：

那时侯摩尔定律还未出现，所以你可以整整十年都在同一个大型机上工作，对吧？就像你不用去考虑下一个操作系统，或者下一个类型的容器编排器，又或者下一个出现 AI 之类的东西一样。你可能会整个职业生涯都在从事 COBOL。而且你知道自己的生活将会非常稳定。

00:04:55 - Saron Yitbarek：

但是，摩尔定律最终还是来了。新的基础设施也出现了。现如今，程序员不太可能去学习一种半个世纪前的旧语言。但实际上，那些老旧的大型机其实并没有消失。这意味着我们对 COBOL 开发人员的需求也没有消失。

00:05:17 - Chris Short：

寻找 COBOL 开发者变得越来越困难。最终会发生的事情是这些大型机可能已经存在了 50 年。这些仍然可以编写出色 COBOL 程序的开发人员将获得巨额收入，以帮助项目运行并完成大型机中的数据重组。而且，该技能肯定会绝迹，并且正在成为一个利润丰厚的职业领域，如果你……现在写 COBOL 绝对可以赚很多钱。

00:05:49 - Saron Yitbarek：

尤其是在制造业和金融业。你无法超越几十年前建立的所有基础设施。遗留代码遍及全球。忽略这些老旧的基础设施及其相关的语言，将是一个巨大的错误。

00:06:08 - Chris Short：

有两千亿行代码摆在那里，要重构这些代码真的很难。不，我不认为在有生之年我们会看到它消失，真的。

00:06:21 - Saron Yitbarek：

Chris Short 是红帽的首席产品营销经理。

00:06:28：

我想花一秒钟解释一下 Chris 的观点。你想想看，95% 的 ATM 交易中都有 COBOL 代码，那就是我们与这种语言的联系。但是，COBOL 程序员的平均年龄并不比该语言年轻多少。他们 45 岁，或许 55岁。新人们并不感兴趣这门语言。这就是为什么我想向你介绍一个人。

00:06:56 - Ritika Trikha：

嘿，我是 Ritika Trikha。

00:06:59 - Saron Yitbarek：

Ritika 是一名技术编辑，曾在 HackerRank 工作。她对 COBOL 的这个问题着迷：人们认为 COBOL 是后大型机时代无意义的残留品。

00:07:12 - Ritika Trikha：

如今的开发人员根本不会考虑 COBOL 了，见也没见过，想也没想过。

00:07:17 - Saron Yitbarek：

但这可能是灾难的根源。

00:07:21 - Ritika Trikha：

如今，仍然有大量的 COBOL 代码在驱动企业的业务。每年至少新增 15 亿行 COBOL 新代码。我认为当你看特定行业时，真的很有意思。

就像美国国税局有 5000 万行代码。社会保障局有 6000 万行代码。因此，这些单位和实体正在处理一些如今最敏感、重要的信息，如果我们不继续为这些大型机提供支持和维护，就会造成很大的破坏。

00:08:04 - Saron Yitbarek：

因此，如果我们无法摆脱旧的基础设施，又无法挥舞魔杖来重建整个大型机业务，我们该怎么办？编码人员有时候仅考虑未来，该如何接受过去？我们首先需要直面该问题。

00:08:25 - Ritika Trikha：

你知道，年轻一代将不得不重拾这些技能。或者，必须对这些大型机进行某种现代化改造。无论是哪种方式，这个问题都不会消失。这就是为什么 COBOL 还活着的原因。

00:08:35 - Saron Yitbarek：

这并不容易。 Ritika 认为我们已经忽略这个问题太长时间了。

00:08:42 - Ritika Trikha：

这非常昂贵、艰巨，并且替换数十亿行 COBOL 代码的风险也非常高。它是用于关键任务的代码，比如社会保障和金融信息。COBOL 是专门为此类大量交易而设计的。因此，它由格蕾丝·赫柏在 60 年代为商业交易而设计。自上世纪 60 年代以来，一直存在“如果没坏，为什么要修复它”的说法，现在我们处于这样一个关头，即延续了数十年的大量的高价值数据运行在 COBOL 上。

00:09:22 - Saron Yitbarek：

从某种意义上说， Ritika 在呼吁一种文化的转变。改变对 "进 "与 "退 "的态度。由于发展的世界慢慢有了越来越久的历史，我们会更加地接触到自己的历史。你无法摆脱老化的基础设施。这意味着你也不能忽略编程语言的历史。

00:09:47 - Ritika Trikha：

有些事情必须得做。当我在 HackerRank 时，我亲眼看到了多少银行和金融机构对 COBOL 开发人员的伤害，几乎是绝望的。这不是一个会被解决的问题，我认为要么必须有某种现代化的系统，要么我们继续培训人员并激励他们。我个人认为将会有 COBOL 再次出现的一天。真的，当所有拥有 COBOL 知识的开发人员退休，并且没有新一代的开发人员学 COBOL 时，将会发生什么？总得做点什么，对吧？所以，当从 COBOL 转向新的基于云的基础设施时，需要有更多的系统化和制度化的改变。

00:10:37 - Saron Yitbarek：

Ritika Trikha 是一名旧金山的技术作家。

00:10:49 - Saron Yitbarek：

那么 Ritika 提到的那些基于云的基础设施呢？我们今天建立的基础设施是否会将后代绑定到特定的语言，像我们仍绑定找 COBOL 上一样？亚马逊 Web 服务（AWS）可能是最大的单一云基础设施，于 2006 年推出。Google 云平台（GCP）于 2008 年问世，微软 Azure 于 2010 年问世。 Go 语言以并发为重点，定位于在新的云基础设施上蓬勃发展。这是这个时代的语言。

00:11:26 - Carmen Andoh：

嗨，我叫 Carmen Andoh， 我是谷歌 Go 团队的项目经理。

00:11:34 - Saron Yitbarek：

Carmen 对 Go 语言与今天的基础设施有怎样的联系有深入的理解。这要从 Go 的创作者和编程语言历史的紧密联系说起。

00:11:47 - Carmen Andoh：

Robert Pike、Robert Griesemer 和 Ken Thompson。这些名字算是从上世纪 60 年代就开始出现了。Ken Thompson 发明了 B 语言，然后他在夏天的假期继续发明 UNIX 操作系统。Rob Pike 发明了字符串编码 UTF-8，他还发明了 ASCII。他帮助 Ken Thompson 共同编写了 UNIX 编程环境。所以，这两个人是很多、很多年前的同事，他们一

直在研究和发明用以前的编程语言编写的操作系统，这些语言包括 Ken Thompson 最终帮助 Dennis Ritchie 一起编写的 C 语言。

00:12:28 - Saron Yitbarek：

Pike、Griesemer 和 Thompson 在 Google 工作之后，他们发现了一个严重的问题。并没有出现大规模的并发。人们等待了几个小时编译出来。他们使用的是 C++，并且必须得编写所有这些回调和事件调度器。那是在 2009 年，我们的基础设施再次发生了变化。诸如 C++ 之类的语言越来越不适应这种新的现实。

00:12:59 - Carmen Andoh：

多核处理器、网络系统、大规模计算集群和 Web 编程模型等正在引入这些问题。而且，还有这个行业的增长，程序员数量在 2010 年就会达到成千上万。因此，直到那时，所有的编程语言都是在规避问题而不是在正面解决问题。

00:13:24 - Saron Yitbarek：

最终，将达到一个临界点，必须开始改变。

00:13:30 - Carmen Andoh：

嘿，我们讨厌 C++ ，我说：“好吧，让我们看看我们是否能发明些新的东西。”

00:13:37 - Saron Yitbarek：

这种新语言需要完美地适应我们最新的基础设施。

00:13:43 - Carmen Andoh：

2005 年云技术到来以后，你不再需要自己的计算机，在某种程度上在其他地方租用它，你就可以得到一个分布式系统。但是在分布式系统中，以及在云计算中，存在并发消息传递问题。你需要确保采用异步对你来说没有问题。Go 缺省就是异步的编程语言。基本上，这意味着你执行的每个操作（例如将所有这些不同的消息发送给系统中的另

一个计算机）都无需等待另一个机器对你的响应即可完成。因此，它可以在任何给定时间处理多个消息。

00:14:28 - Carmen Andoh：

就是说，云计算是分布式的。因此 Go 的开发就是来满足这一确切需求。Go 早就成为进行这种分布式计算的标准方法之一。这就是为什么我认为它立即引起了开发人员的广泛关注。Go 绝对是云基础设施的语言，无论是其设计，还是所有云基础设施工具，以及在过去十年中如雨后春笋般出现的模块的生态。

00:15:06 - Saron Yitbarek：

很快，诸如 Kubernetes 之类的关键应用都用 Go 编写了。谷歌还创建了 Go Cloud，这是一个开源库和一系列工具，使得 Go 更具吸引力。很显然，它是新生态系统的语言。它是云的语言。而且，它的创造者们因开发生命力持久的语言而享有声誉，这绝对没有坏处。

00:15:33 - Carmen Andoh：

我认为业界的其他人会说：“嘿，我认为这不会很快消失。”这种语言的发明者恰巧也发明了语言有 50 、60 年了。

00:15:47 - Saron Yitbarek：

Carmen Andoh 是谷歌 Go 团队的项目经理。

00:15:54：

因此，我们有了一种新的语言 Go ，旨在提供云基础设施必需的并发性。听起来不错。Go 的设计师倾向于创造可以持续半个世纪的语言。这也很棒。但是我的问题是，从现在起，50 年后，当 Go 更像是 COBOL 时，这到底意味着什么？当世界上充满了只有老开发人员才能理解的旧版 Go 代码时，这又意味着什么？在当今的云基础设施老化的时候，我们是否会做好准备？我们是否从 COBOL 和大型机领域吸取了教训，可以帮助我们为 Go 和云设计更美好的未来？

00:16:40：

幸运的是，我找到了问所有这些问题的合适人选。这就是下面这位。

00:16:51：

我们如何使我们的语言能面向未来？我们知道他们与当今的基础设施息息相关。我们也知道，随着数十年的发展，新的基础设施必将取代旧的基础设施。那么，我们今天做些什么以确保将来能平滑演进？

00:17:10 - Kelsey Hightower：

我是 Kelsey Hightower ，我在谷歌，是一名开发人员推广大使，我致力于引入开放性技术并将它们应用于谷歌云上的产品。

00:17:19 - Saron Yitbarek：

Kelsey 花了大量时间思考编程的未来。我很好奇，是否有一天我们将陷于握有 Go 语言技能的是另一批老龄化的程序员的问题，就像我们现在缺少 COBOL 的引导一样。我们是否在为这个长远的未来做计划？因此，我和 Kelsey 坐下来进行讨论。

00:17:42 - Kelsey Hightower：

...等等。但是，如果你考虑到今天面临的一些新的挑战，如应对互联网，这种网络，你将面临许多用户，成千上万的并发用户，以及不同的机器和架构类型的组合。考虑到这些新的场景，因此你通常希望有一种新的语言来解决。例如， JavaScript 是用于 Web 的，你不会想改造 COBOL 以便可以用它来进行 Web 编程。最终，我们今天已经有了数百种相当完善的语言，而且它们都非常专注于他们的优势。

00:18:15 - Saron Yitbarek：

那么，在那种情况下，我们是否需要积极推动人们走向 COBOL ？如果我们正在为这些新问题开发新语言，并且它们是高度定制化的，而 COBOL 仍在坚持不谢幕，我们是否需要鼓励人们选择它，以便我们可以维护我们的旧代码？

00:18:32 - Kelsey Hightower：

好吧，我认为这将对企业是个挑战吧？所以，你已经在 COBOL 上投入了 10 到 20 年，没有人会主动想学习一些新的 COBOL。或者你不会像刚从大学毕业那么“我要加倍努力……”。

00:18:45 - Saron Yitbarek：

没错。

00:18:46 - Kelsey Hightower：

“...这种语言比我父母的年龄都大。” 因此，在那个领域，你必须问自己，继续使用 COBOL 的风险是什么？未来是否仍然有意义？我认为它仍然有益于某些类型的工作任务，但是我们必须问自己一个问题，是时候推进了吗？是时候进化一点了吗？因此，如果你仍然有数十亿行的 COBOL 代码，那么你将必须寻找所有剩余的 COBOL 人才，并将其招进来，但也许我们该开始考虑其他语言能从 COBOL 学习些什么，并将某些功能和库加入到其他语言中。

00:19:26 - Saron Yitbarek：

COBOL 终止以后的日子，将会是一个巨大的基础设施项目。用我对纽约地铁的比喻，就像是要替换每条地下隧道。因此，展望未来，我想知道我们是否可以预见到这些问题，甚至将来对自己也能有所帮助。

00:19:48：

如果我们将今天的云计算与大型机进行比较，我们是否会在最终到达同一条船上，有着这些旧代码库，使用着旧的但非常稳定的语言，而且我们也到了必须做出选择的时候，我们应该继续前进还是保持不变？

00:20:05 - Kelsey Hightower：

云有点不同的是它不是来自一个制造商，对吗？许多云提供商通常提供了一系列技术集合，你就可以选择编程语言、编程范式，无论你是要事件驱动还是基于 HTTP 的全 Web 服务。这意味着你可以选择编

程的方式，然后只需专注于要解决的问题。因此，数据进进出出，但是你来选择处理数据的方式。

00:20:36：

大型机通常只有一个主界面，对吗？就像你编写一份任务一样，这就是你提交任务的方式，这就是你监控任务的方式，这就是结果。这本身就是一个限制。如果你考虑一些较新的大型机，它们也会支持些较新的技术，因此即使在大型机领域，你也会开始看到可用于运行任务的编程语言扩展。

00:20:58：

那么我们开始问自己，好吧，既然我有了新的选项，该什么时候离开这种特定的编程范式继续前进呢？我认为我们不会陷入困境。

00:21:08 - Saron Yitbarek：

正确。

00:21:08 - Kelsey Hightower：

我认为新的分布式机器很不错，可能起步成本更低，你不必购买整个大型机即可开工。但是我们仍然希望易用性和之前一样：给你我的工作，你为我运行它，完成后告诉我。

00:21:24 - Saron Yitbarek：

绝对是这样，你觉得发生的事情，或者说 COBOL 所发生的事情，会发生在今天的任何一种语言上吗？ 以 Go 语言做例子，你看到我们在努力地改进 Go 从而吸引人们在 30 年后还想用 Go ？

00:21:38 - Kelsey Hightower：

我认为所有语言都会遭受这种命运吧？如果你仔细想一下，其实 Python 已经存在很长时间了。我想差不多 20 年了，甚至更久。因此，我认为会 …… Python 重新流行起来了，它现在是机器学习的基础语言。

00:21:53 - Saron Yitbarek：

是的。

00:21:54 - Kelsey Hightower：

对于 Tensorflow 之类的库，如果我们仅用时间来衡量，我认为这可能不是看待它的正确方式。还应该有社区是否活跃？语言的适配意愿是否活跃？我认为 Python 做得确实非常出色……社区看到了使其他语言更易于使用的能力。例如，Tensorflow 底层有很多 C++ ，使用这种语言编程可能没有 Python 这样的用户友好性。你可以用 Python，并用它来生成人们正在使用的一些东西，例如 Tensorflow 。现在机器学习非常热门，人们将 Python 引入了这个新领域，那么你猜怎么着？Python 仍然是活跃的，并且在未来的一段时间内都是活跃的。对于 Go 来说，这同样适用。如果 Go 能够继续保持活跃度，那么，它就像许多基础设施工具和许多云库的基层一样，它也将保持活跃。因此，我认为都是这些社区确保了它们将来占有一席之地，并且当未来出现时，确保那里仍有它们的位置。

00:22:58 - Saron Yitbarek：

是的。那么，我们如何让我们的语言面向未来呢？就是说，我们如何有意地设计一种语言，使其能持续存在，并在 20、30 年内都保持活跃呢？

00:23:10 - Kelsey Hightower：

使用语言的人，我认为这在开源世界中确实是独一无二的。现在，我们已经不再使用商业语言了，对吧，过去曾经来自微软的语言或太阳微系统的如 Java™ ，那时侯，每个人都依赖于供应商来尽心尽力来对语言能干什么以及对运行时环境进行新的改进。现在，我们看到的诸如 Go、Node.js、Ruby 之类的东西，所有这些都是社区支持的，并且社区专注于运行时环境和语言。这样任何人都可以添加新库，对吧？有一个新的 HTTP 规范，对，HTTP/2 几年前问世了，每个社区都有贡献者添加了那些特定的库，猜猜现在怎么样？所有现在这些语言，都兼容于大部分的未来网站。

00:24:01：

我认为现在是个人真正地拥有了更多的控制权，如果他们想让语言适用于新的用例，只需要自己贡献即可。因此，我们不再受限于一两家公司。如果公司倒闭，那么运行时环境可能会因此而消亡。我们不再有这个问题了。

00:24:23 - Saron Yitbarek：

我们之前在这个播客上已经说过了。未来是开放的。但是，令人着迷的是考虑怎样能做到再过几十年，过去也将是开放的。它们将继承能够变形和演进的基础设施和语言。

00:24:39 - Kelsey Hightower：

太棒了，感谢你的加入，我期待人们的工作，而且大型机仍然有意义。它们是经典技术，因此我们不称其为遗产。

00:24:47 - Saron Yitbarek：

哦，我喜欢，经典，非常好。

00:24:51：

Kelsey Hightower 是 Google 的开发者推广大使。

00:24:57：

我正在想象一个充满经典编程语言以及尚未诞生的新语言的未来。那是我为之兴奋的未来。

00:25:07 - 演讲者：

请离关着的门远一点。

00:25:12 - Saron Yitbarek：

要知道，2017 年 Andrew Cuomo 州长宣布纽约市地铁进入紧急状态。他的政府拨出 90 亿美元投资于老化的基础设施。这应该提醒我们，迟早我们得注意遗留的系统。你不仅需要继续前进，面向未来。你还背着历史包袱。

00:25:37：

在开发的世界中，我们倾向于偏向未来。我们认为我们的语言仅在它们流行时才有用。但是，随着信息基础架构的不断老化，开发的历史变得越来越真实。事实证明，过去根本没有过去。记住这一点是我们的工作。

00:26:05：

你可以前往 redhat.com/commandlineheroes，以了解有关 COBOL 或 Go 或本季我们要介绍的其他语言的更多信息。那里有很多很棒的材料在等你。

00:26:19 - Saron Yitbarek：

下一集是关于 Bash 的。我们将探索 shell 脚本的起源以及自动化的关键。

00:26:30 - Saron Yitbarek：

《命令行英雄》是红帽的原创播客。我是 Saron Yitbarek 。下期之前，编码不止。

什么是 LCTT SIG 和 LCTT LCRH SIG

Special Interest Group

LCTT SIG 是 LCTT 特别兴趣小组，LCTT SIG 是针对特定领域、特定内容的翻译小组，翻译组成员将遵循 LCTT 流程和规范，参与翻译，并获得相应的奖励。LCRH SIG 是 LCTT 联合红帽（Red Hat）发起的 SIG，当前专注任务是《代码英雄》系列播客的脚本汉化，已有数十位贡献者加入。敬请每周三、周五期待经过我们精心翻译、校对和发布的译文。

欢迎加入 LCRH SIG 一同参与贡献，并领取红帽（Red Hat）和我们联合颁发的专属贡献者证书。

---

via: https://www.redhat.com/en/command-line-heroes/season-3/the-infrastructure-effect

作者：Red Hat 选题：bestony 译者：messon007 校对：wxy

本文由 LCRH 原创编译，Linux中国 荣誉推出