白皮书 - 2020 年 4 月

利用 VMware Tanzu Application Service 进行 Spring 开发指南

vmware

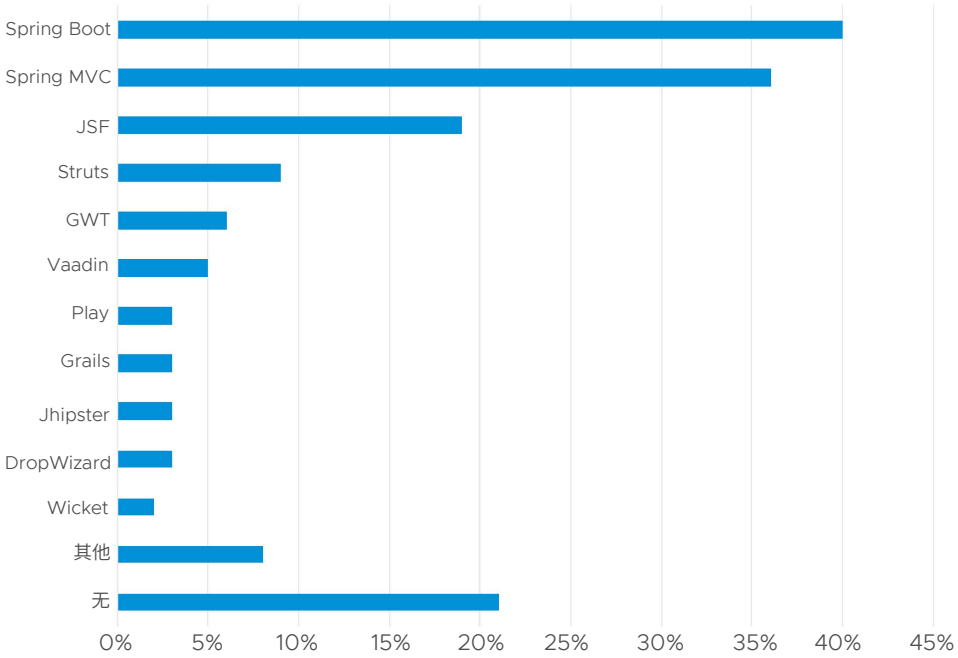
目录

Pivotal 现已归入 VMware 旗下，因此，其中一些产品和服务现已成为 VMware Tanzu™ 的一部分。了解更多。

简介

“无需多言，这张图表很好地解释了 Spring 在 Java 生态系统中的主导地位。10 名开发人员中就有 4 人在其应用中使用 Spring Boot，它已首次超过 Spring MVC 框架，这是一个很有意思的现象。”

SNYK AND JAVA MAGAZINE [1]



这份白皮书的读者很有可能就在使用 Spring Boot。它是用于在云原生 Java 中构建应用的出色框架。现在，您可能在使用许多 Boot 应用。您可能会想“在哪里运行这些应用最好？”

不要轻易做这个决定。如果做出正确的选择，您的 Java 虚拟机 (JVM) 就能实现超音速 - 您可不断地快速交付高质量软件。事实上，您利用 Spring Boot 实现持续交付的能力决定了许多事情的成败。想想企业面临的处境吧。

新的初创企业利用新奇的技术追逐您的客户。超大规模 Web 公司一直对新市场虎视眈眈。同时，您也在努力寻找在业内成为颠覆者的方式，并希望使用软件在竞争中脱颖而出，赢得市场份额。

好消息是，像您一样的数百家企业已利用 Spring Boot 和 VMware Tanzu Application Service (TAS) 正面应对这一挑战。您已经知道了 Boot 的精彩之处。那么 Tanzu Application Service 又有什么特别之处呢？它又为什么能够如此出色地运行 Boot 应用呢？

1. Snyk and Java Magazine。“JVM 生态系统报告 2018。”

目标、定义和范围

请把本白皮书当作参考指南。您在各个章节和整个文档中均可发现价值。

我们的目标是探索在 Tanzu Application Service (TAS) 上运行 Spring Boot 微服务这一组合的强大威力。很少有公司希望减缓发展速度。但速度慢是与风险过多、选项不足相伴而来的副产品。Tanzu Application Service (TAS) 和 Spring Boot 的组合能够尽可能降低风险，并最大限度地提升工作效率。

在本白皮书中，我们将证明这一观点。我们的方法是什么呢？首先，我们要构建一小组应用。然后，我们要将所有应用推送到我们选定的云平台 TAS。

跟我们一起了解一些知识，并收获一些乐趣吧。

到底什么是应用平台？

应用平台是大规模运行定制代码的常用方式。TAS 就属于此类解决方案。企业开发团队使用 TAS 和其他应用平台处理整个软件开发生命周期。

“应用”这个术语有何意义？在应用平台中，可部署的构件通常是经过编译，可部署的应用库。对于 Java，您会推送 *.jar 文件；对于 .NET，您会推送 *.dll 或 *.exe。因此您可以将应用推送到平台，而不必使用容器镜像，工具会为您完成其他工作。

而其他工作都包含哪些内容呢？现代应用平台会代替您做以下工作：

- 网络和路由 - 可为您完成负载均衡、DNS 和其他网络连接功能
- 日志、指标和监控 - 应用遥测可帮助您观察应用并进行故障排除
- 服务市场 - 为您的应用添加支持服务的结构化方式
- 团队、配额和使用情况 - 有用的监管功能可帮助您遵守常见的企业租户要求

所有这些要素最后组合成开发和运维团队的完整体验，我们称之为“平台”。

出色的应用平台是否会使用容器？当然。事实上，应用平台会为您生成容器。对于 TAS 来说，它使用常用的生成包模式将应用容器化。这种方法可帮助开发人员，因为您不必为容器的生命周期操心。（实际上，如果愿意，您可以在 TAS 上运行自己的 Docker 或 OCI 合规容器镜像。我们在另一篇文章中做了介绍。）

TAS 的最终目标是让您专注于编写可大规模顺畅运行的代码。

现在您已对应用平台有了更多的了解，让我们再来谈谈微服务。

微服务探微

在钻研代码之前，让我们先来了解一下微服务。您如何希望成功开发云原生软件，这种模式必不可少。

微服务本身并不意味着确定的框架、语言或运行时。您掌握着很大的自由度。因此不难理解，小型开发团队（初创企业）能够很好地利用微服务。但大型企业呢？微服务真的能随数千名开发人员进行扩展吗？不难想象微服务带来巨大混乱的情况。

企业级微服务

如果您需要稳定、灵活、安全并且可扩展的体系架构，那么大规模微服务值得考虑。在选择微服务之前，请考虑这些问题：

- 您的系统是否需要以不同的速度发展，或向不同的方向发展？
- 某一模块是否有完全独立的生命周期？
- 系统的某一部分是否需要独立扩展？
- 系统的某一部分是否需要故障隔离？
- 系统的某一部分是否需要与外部组件隔离？
- 您是否需要自由地选择适当的技术来完成任务？
- 您的开发人员是否会浪费时间等待其他人来清除障碍？

在这些问题中，如果有肯定回答，那么即使微服务会带来额外的复杂性也值得使用。

如果您认为微服务值得一试，如何在采用这种模式时最大限度地降低复杂性？您需要数字化框架来使微服务正常地发挥作用。如果您在自动化的现代平台上运行微服务，成功的几率要大得多。您可以问问同行。

像 TAS 这样的平台包括防护措施，即使服务不断变化，也可确保它们在平台上顺畅运行。TAS 还对其行为做出了一些承诺。开发人员、运维人员、InfoSec 专业人士和架构师发现这些承诺（以及相应的可预测性）很有用。

让我们仔细斟酌一下什么是重要的事，然后再继续。最重要的是您自己。您的业务、您的应用理念和您对客户的影响力都烙刻于软件工件中。请关注这一点并进行大量投入。您应在微服务、应用和现代开发实践上投入。可以利用框架和平台来处理基础架构和其他无差别的繁重工作。毕竟对您的业务而言，最为重要的是快速交付您的客户喜爱的高质量软件。

现在，让我们看看这组 Spring Boot 应用。

Spring Boot 启动：快速入门

基准应用

自从 Rod Johnson 于 2003 年创建 Spring 以来，它已获得了长足发展。Spring Boot 现已发展到第二代，可提供有用的预配置 Bean 用于核心抽象，例如 Web 服务器、安全性、数据访问和连接。Spring Boot 的出现无异于一剂催化剂，它使 Spring 成为在 JVM 上进行开发的不二之选。

现在，Java 开发人员可利用由工具、项目和基于 JVM 的语言构成的丰富生态系统构建云原生应用。针对配置方法的约定极受欢迎，可帮助企业开发人员完成编写出色软件的任务，同时无需进行繁琐的前期配置。

关于 Spring Bean

Spring Bean 有时也直接称为 Bean，是指由 Spring 管理和创建的一段代码。例如，您可能有一段用于管理购物车功能或帐户设置的代码。您可以将它们称为 Cart Bean 和 Account Bean。Spring 开发人员经常会讨论他们的 Spring Bean。Spring Boot 可以自动配置您的应用，并创建 ApplicationContext，以及一组由 Spring 管理的 Bean。这种构建应用的选项式方法可使开发人员不必手动配置 servlet 容器、数据源和安全要素等内容。

介绍了这些背景之后，让我们从应用开发人员的角度更深入地探索 Spring Boot 的细节。非开发人员也能从中有所收获，特别是在您所选云基础架构的 TAS 中运行这些示例的情况下。

首先，我们从一张图片开始讲起，我们要构建的是什么？下图展示了我们的一组基本微服务。我们会在此基准的基础上陆续添加功能。压轴节目是将它们全部推送到 TAS。

大家可以看到，我们有一个简单的前端 UI 以及后端 API，即以下组件：

- 在 Vue.js 中实施的 Todo UI（改编自 todomvc.com）
- Spring Cloud Gateway 中的 Todo Cloud Gateway
- 在 Spring Boot 中实施的 Todo API

让我们详细介绍一下每个应用。

Todo Cloud Gateway

Todo Cloud Gateway 是一款 Spring Boot 微服务，可作为 Todo 微服务的 API 网关和路由器。它以 Spring Cloud Gateway 为基础，默认运行 application.yml 中的路由定义的网关。当 Todo 网关在 Spring Cloud 环境中启动后，它会与服务发现进行同步，并且当微服务开始在线时对其进行动态路由。与此类似，它也会删除离线的微服务路由。（稍后我们会进一步讨论 Spring Cloud。）

Todo UI

Todo UI 是 todomvc.com 的 Vue.js 实施。我们将使用它通过 Todo Cloud Gateway 与 Todo API 直接交互。这是此版本与 todomvc.com 上的版本的主要区别。我们的特色是调用后端 API 创建、检索、更新和删除 Todo。我们利用 VueResource 执行此操作，它进行 HTTP 调用以创建、读取、更新和删除 Todo。（因此与 todomvc.com 上的 Vue.js 应用相比，此处有一些不同的日志语句。）

TAS：灵活的应用平台，可支持一系列框架

Todo UI 只使用了 HTML、JavaScript 和 CSS。它并不是基于 JVM 的。这展示了 TAS 的另一个属性：该平台可支持广泛的应用。我们相信，每个应用都可以得益于现代平台。TAS 可连接前端和后端开发人员，还支持您使用喜爱的框架交付全栈软件。

Todo API

Todo API 展示了使用 Spring Boot 生成由 API 驱动的微服务是多么容易。如果您刚刚开始使用 Spring Boot，那么 Todo API 就是值得您学习的一种有用方式。它可为 Todo UI 实施一个 API 后端。默认情况下，API 会把 Todo 保存在映射中，其大小取决于 todos.api.limit 属性。（在下一节中，我们将了解如何使用 Spring Cloud Config 服务器更改大小限制。）

对类进行 @RestController 和 @RequestMapping 注释后，我们可以进行封装并为 API 提供上下文。Todo API 将根目录环境中的 HTTP 请求映射到 CRUD 方法。Todo Data 微服务会公开一个类似的 CRUD API，但我们无需编写代码。它会使用 Spring Data Rest，凭借 CRUD API 覆盖数据模型。查看 Todo Data，了解有关 Spring Boot 的更多信息，以及 Spring Data Rest 如何将您的数据模型公开为一组 REST 端点。

API 操作

TodosAPI 类包含的方法可从 TodosUI 进行调用。大家可以看到，定义 RequestMappings 来处理 HTTP 请求非常简单。类似地，使用 Spring Boot 2.0 的被动式堆栈定义相同的 RequestMappings 作为非阻塞端点，也同样简单。

不熟悉被动式堆栈？真是太巧了。

如果您是第一次听说被动式堆栈，正好听我们介绍一下。虽然被动式开发模式的前提已存在了一段时间，但现在我们才更大规模地将其用于实践。被动式体系架构使应用能够以异步方式进行响应。这样数据的创建者和用户就能各自独立地进行扩展。在您的职业生涯中，可能使用过某种消息收发系统。这听上去很熟悉，是吧？现在假设您的应用代码也是以事件响应的理念编写的。那么它可能也会被归为被动式应用。

Spring MVC	Spring WebFlux

`@PostMapping("/") public Todo create( @RequestBody Todo todo) { }`
 	
`@GetMapping("/") public List<Todo> retrieve() { }`
 	
`@GetMapping("/{id}") public Todo retrieve( @PathVariable Integer id) { }`
 	
`@PatchMapping("/{id}") public Todo update( @PathVariable Integer id, @RequestBody Todo todo) { }`
 	
`@DeleteMapping("/{id}") public void delete( @PathVariable Integer id) { }`
	`@PostMapping("/") public Mono<Todo> create( @RequestBody Mono<Todo> todo) { }`
	
	`@GetMapping("/") public Flux<Todo> retrieve() { }`
	`@GetMapping("/{id}") public Mono<Todo> retrieve( @PathVariable Integer id) { }`
	
	`@PatchMapping("/{id}") public Mono<Todo> update( @PathVariable Integer id, @RequestBody Mono<Todo> todo) { }`
	
	`@DeleteMapping("/{id}") public Mono<Todo> delete( @PathVariable Integer id) { }`

现在我们已介绍了基本部分，接下来是更加有趣的内容 - 我们要把应用推送到 TAS！

该把代码推送到 Tanzu Application Service 了

开始之前，请确保您已安装了 cf-cli 并满足一些简单的先决条件。

首先我们需要：登录到 TAS。（需要加快速度？我们推荐使用 Tanzu Web Services - 这是很棒的工具，还可以免费试用。）

除了代码之外，我们只需要一个清单和一个可选的 vars 文件。这两个构件用于描述我们即将上线的云应用，后续在生产环境中运行时，它们都会带来极大的优势。请参阅此存储库，了解有关将这些应用推送到 TAS 的更多信息。

vars 文件

我们的 vars 文件包含部署的动态元素。我们会将可能更改的内容详情放在这里。例如，我们包括了每个应用的公共路由以及相应的应用构件。以下是开发人员非常感兴趣的一个细节：我们的应用将以原生形式推送到 TAS 中。TAS 会融合这些应用，并实际运行它们。任何以开发人员为中心的平台都应在源和软件之间提供一层极薄的“大气层”，这就是 TAS 的作用。

以下是 vars 文件的示例，它针对我们的推送定义了属性值。

```
api:
    route: https://todos-api.cfapps.io
    artifact: todos-api/target/todos-api-1.0.0.SNAP.jar
ui:
    route: https://todos-ui.cfapps.io
    artifact: todos-ui/

gateway:

    route: https://todos-cloud-gateway.cfapps.io
    artifact: todos-cloud-gateway/target/
    todos-cloud-gateway-1.0.0.SNAP.jar
```

清单

清单对我们的应用进行简单定义。这些文件易于通过配置管理控制和更新。

现在让我们创建一个清单，声明如何部署我们的 3 个应用。vars 文件中的变量注入占位符 (()) 中。这 2 个是我们需要推送的文件。TAS 通过清单知道如何将您的应用容器化。（这是由生成包属性定义的。）生成包可为容器提供必要的应用运行时（例如 JVM 或 NGinx 服务器）。希望进一步了解生成包？请查看此概述和我们的技术文档。Java 生成包是将基于 JVM 的工作负载交付到 TAS 的稳健方式，可满足您的需要。

```
part-1-manifest.yml - app deployment
---
applications:
-name: todos-api
     routes:
     -route: ((api.route))
     path: ((api.artifact))
     buildpack: java_buildpack
-name: todos-cloud-gateway
     routes:
     -route: ((gateway.route))
     path: ((gateway.artifact))
     buildpack: java_buildpack
     env:
          TODOS_UI_ENDPOINT: ((ui.route))
          TODOS_API_ENDPOINT: ((api.route))
```

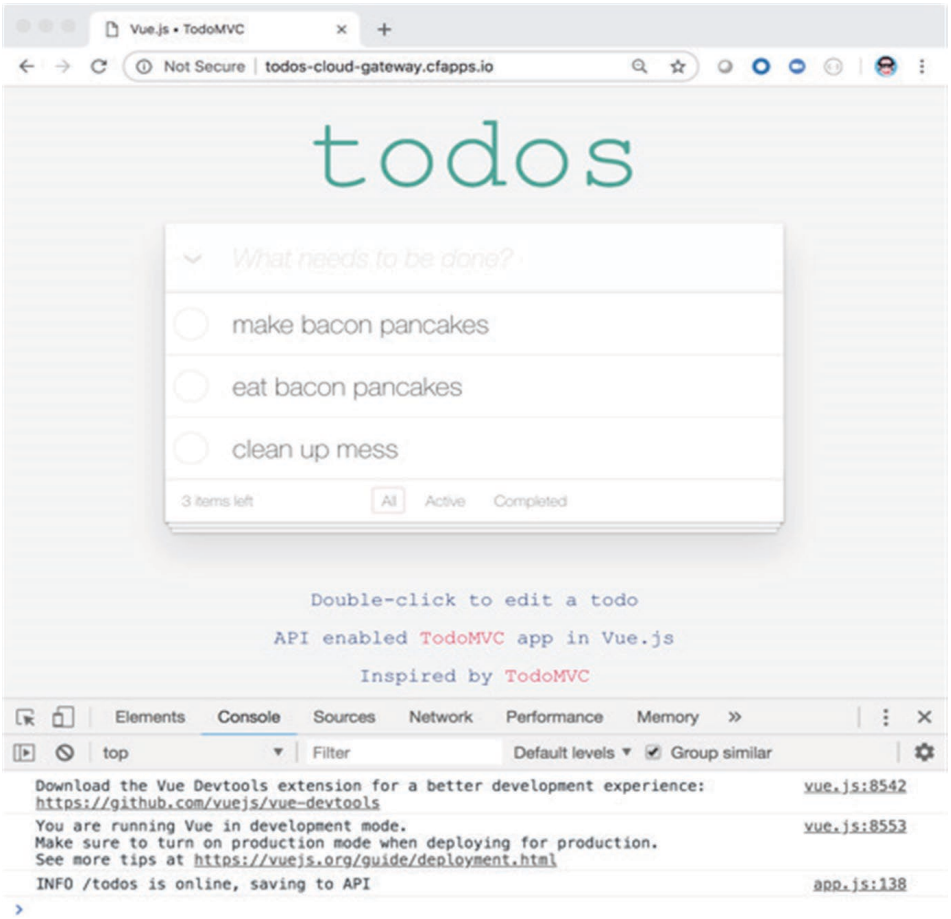
现在，我们要使用云中最具魔力的命令了，它就是：cf push。

```
cf push -f part-1-manifest.yml --vars-file part-1-vars.yml
```

这条命令会使用上述生成包抽象，在 TAS 上启动一项任务，将我们的应用容器化。TAS 会处理我们的应用，并在 droplet 中部署适当的操作系统和应用堆栈。该平台然后会启动我们的应用。TAS 和 cf push 命令还有许多实用功能，例如：

- 为您创建安全、强化的容器，无需不必要的操作系统
- 为您的应用设置路由（或 URL）
- 为您的应用创建负载均衡条目
- 为您的应用创建 SSL 终止
- 为您的应用创建运行状况监控和日志记录子系统
- 以良好的运行状态启动您的应用，具有理想的实例数量
- 注入服务凭证，为您的应用绑定所需的任何支持服务

推送完成后，我们可以在浏览器中提取 Todo UI，并制作重要待办事项列表。



请花点时间回想一下我们刚刚所做的事情。我们从零构建了在云中部署的全栈应用，功能已非常接近完整。（是的，我们还需要添加单点登录。但这里就不演示了。）

“JVM 上的速度是大杀器”

我们已经构建出一个有用的 Spring Boot 应用。而且我们没费多大力气就将其推送到生产环境。但这就是我们的目标吗？ Netflix Cloud 的 Andy Glover 2014 年在 SpringOne2GX 2014 主题演讲中的精辟表述，正是我们希望达成的目标：

确实如此。JVM 上的速度让我们能够快速发布代码，获得经验教训，然后根据反馈发布更多代码。这是一个良性循环，也是企业在云时代获得成功的生命线。

明智地选择运行 Java 应用的位置

有许多不同的方式可以运行 JVM 工作负载。如何才能获得最佳速度？理想的运行时应该能够：

- 将应用上线必须完成的许多任务自动化
- 提供简单、一致、可配置的开发体验
- 自动执行容器化
- 出现问题时提供稳健的工具和方法进行故障排除
- 提供一致的可配置运维模式
- 支持在不同的云基础架构环境中部署应用

这很像 TAS，不是吗？让我们与基于 JVM 的工作负载的其他运行方式进行比较。假设我们有一组相同的基准 Todo 应用。如何才能获得与 TAS 相同的效果？

请考虑基于 JVM 的应用的多种启动方式。您可使用各种技术和规则传递 JVM 和应用参数，缺乏统一性和标准化。这样就很难加快速度。

让我们再来看看 Todo UI 前端应用。它需要 NGinx 这样的 Web 服务器。在全栈环境中，可能要花费大量时间和精力配置并维护简单的 Web 服务器。利用 TAS，我们可以在 NGinx 等现成 Web 服务器中轻松部署前端 Web 应用，由这些 Web 服务器完成路由和负载均衡的工作。

那 Docker 容器呢？当然，人们会用它搭建正确的操作系统和中间件堆栈。当所有应用都拥有了适当的 Dockerfile 后，您需要构建每个容器镜像并上传到容器存储库。实施一致的容器化本身就是一项巨大的投资，只要问问尝试过的人就可以知道。此外，这还是 InfoSec 的噩梦，因为堆叠容器的方法不计其数。您无法知道某个容器中有什么，以及它的漏洞可能位于何处。因此您会听到人们把手动生成的容器叫做“看不出原料的肉罐头”。容器最好是可扩展的自动化流程的产物。

按捺住亲自构建平台的冲动 - 购买平台并推动业务转型

那些决定运行自己的平台的公司很快就发现一个残酷的现实：这比他们想象的要昂贵。在与大量财富 500 强公司合作的过程中，我们发现，即使最简单的 DIY 平台也需要花费 2 年时间构建，单单工资一项就要花费 1400 万美元（60 名工程师）。而这才刚刚有了最低限度可用的产品。说真的，您的公司能够等待两年时间再开始云原生转型吗？

当然，平台投入生产后，成本还会继续增加。随着体系架构模式和技术的不断发展，对新功能的需求会使平台难以跟上发展节奏。开发人员可能要求评估并增加某些功能，以支持新兴模式，例如使用服务网络或功能即服务。除了新功能，平台运维团队一定还需要管理底层软件组件和操作系统镜像的生命周期。他们需要快速采取行动，以应对任何漏洞威胁，并应用最新的安全补丁程序。

在自己构建的平台中添加这些新功能并确保其安全，需要对工程师团队进行相应投资。过一段时间，您的平台团队就会开始像一家云平台软件公司了，但有一个重要区别：您构建的平台并不会真正为您的核心业务带来收入。它会产生费用，并快速增加技术债务。

而这些开支是不合理的，除非您的核心业务就是销售云平台服务。您不希望最有才华的技术人员去构建平台。您希望他们努力增加独特的业务价值，让您的业务与众不同。幸好 Tanzu Application Service 已经解决了创建充满活力、可扩展的安全企业级平台的问题。

Boeing、Mastercard、T-Mobile、StubHub、Rabobank、Dick’s Sporting Goods、Liberty Mutual 和 Comcast 等客户都在使用我们的云原生平台，以前所未有的速度交付高质量软件，并支持企业创新、蓬勃发展。

当然，您可以利用 Docker、Kubernetes 或自行构建的堆栈实现相同的成效。但 TAS 即时可用的体验独具特色，简单优雅。请记住，我们不仅需要云中的 JVM，我们还需要快速交付出色的软件。TAS 提供出色的平台体验，可按模式进行扩展。

Spring 和 TAS 配合使用，可支持开发人员高效工作，与其他备选方案进行对比更可彰显其优势。

Spring Cloud：扩展功能的简单途径

我们已经了解到，TAS 是功能丰富的应用平台。当您的应用部署规模扩大或速度提升时，TAS 可提供基础架构和一致的 DevOps 体验以及应用环境。例如，观察我们的 Spring Boot 应用可发现，TAS 提供的属性和环境可以很好地互相协调。假设您声明了一个应用属性，例如 todos.api.limit=128。该属性可从 TAS 创建的环境实现外部“水合”，方式如下：

```
cf set-env your-app TODOS_API_LIMIT 128.
```

这是很棒的功能，但不可避免地，随着应用产品组合的发展，您需要的不只是一致的平台，您还需要诸如配置、可发现性、恢复能力和追溯性等云的基本组成要素。当然，我们的目标是提供一种方式，供应用和微服务在数字化生态系统中进行交互，这正是 Spring Cloud 实现的功能，这样您就可以专注于编写重要的代码，而不是样板文件。

Spring Cloud 在易于理解的框架中提供便于上手的构造块，例如配置管理、服务发现、回路保护、路由和跟踪。在 JVM 上进行云原生开发从未如此简单，扩展能力也前所未有。

如需更深入地了解 Spring Cloud，请参考以下资源：

- 适用于 TAS 的 Spring Cloud Services - 创建并管理 Spring Cloud Config 服务器、提供服务发现和回路保护功能，减少您的工作量。
- 适用于 TAS 的 Spring Cloud Data Flow - 自动化置备 Spring Cloud Data Flow，编排 Spring Cloud Stream 和任务，您可更多地关注 Stream 或任务应实现的功能，而花费较少精力关注如何实现它们。
- 扩展的 Todo 示例，包含 Spring Cloud 的出色功能，可直接使用 cf push - 这组示例汇集了更大的 Spring 生态系统中的工具，包括 Spring Cloud Config、服务发现、回路保护、Spring Cloud Gateway、Spring Cloud Sleuth、Spring Cloud Streams、Spring Data JPA 和 Spring Data Redis。这些都是云原生开发人员的必备工具。

在 Tanzu Application Service 上运行您的应用：开发人员需要知道什么

刚才我们介绍了构建应用并推送到 TAS 的体验。现在，我们要考察如何毫不费力地在云中运行我们的应用。

让我们继续把 TAS 作为部署目标。

首先要登录。请记得安装 cf-cli，并满足先决条件。

那么在 TAS 上运行应用需要使用哪些命令？正等着大家提出这个问题呢！现在我们就来介绍。

应用是一等公民

TAS 是应用平台。这意味着 TAS 的价值需要应用来体现。这也许是显而易见的事，但却是关键点。我们与 TAS 进行的每次互动几乎都需要指定应用，以及相关操作。

推送

使用 vars 文件将应用推送至 TAS，可设置推送操作的具体变量。我们的示例中经常会用到 vars 文件。

```
cf push --vars-file vars.yml
```

事件

此命令可列出应用的生命周期事件。你会经常用到它，因为可以很方便地看到您的应用上发生的事件的时间、内容、人员和原因。

```
cf events todos-api
```

SSH

但愿您不需要在容器中使用 SSH。但如有需要，可以很方便、很安全地进入运行应用的容器。

```
cf ssh todos-api
```

使用 SSH 进入容器后，了解一下内部结构很有帮助。默认情况下，您以 vcap 用户身份 (whoami=vcap) 进入 $HOME。列出 $HOME 将显示 /app，这是我们的应用所在的位置。由于大多数示例都基于 Spring Boot，您会发现一个 .java-buildpack 目录。其中包含整个 JVM 中间件堆栈。

启动、停止、重新启动、重新调试

TAS 支持您对应用进行生命周期控制。您可分别启动、停止、重新启动和重新调试每个应用。启动和重新启动都是简单的命令。它们都会应用服务绑定的所有新的环境变量和服务。

```
cf start todos-api cf
restart todos-api
```

重新调试是重建操作。这意味着容器将根据现有的应用和生成包重新创建。与启动和重新启动类似，此操作将应用与应用绑定的新的环境变量和服务。

```
cf restage todos-api
```

生成包

我们之前提到过，生成包为应用提供运行时支持。它们可以检测应用类型、设置所需的应用框架，并将您的代码集成到结构良好的容器中。TAS 将您的应用的原生形式作为输入，并在 cf push 过程中应用生成包。输出则是在平台上运行的容器镜像。

TAS 自带一组系统生成包，可实现源到容器的流程自动化，适用于多种语言，例如 Java、Kotlin、Groovy、.NET、Node.js、Python 和 Ruby。开发人员可以自由地使用首选框架，如果您希望大规模提升速度，这是必不可少的。开发人员有多种选择，精简的代码到容器自动化对于加快速度必不可少，特别是在大型企业中。大公司中有数以千计的应用。这意味着在应用层，框架、依赖关系和安全构件有大量变体。在将这些变体容器化时，生成包有助于实现标准化。这对提升开发人员工作效率极为有利，在运维方面也有诸多好处，特别是对于一致性和合规性而言。

Java 生成包

Java 生成包可将基于 JVM 的应用容器化。它支持一组标准容器组件，例如：

- Java main()
- Servlet 容器：Apache Tomcat
- 应用框架，例如 Spring Boot、Ratpack 和 Play

其他应用框架组件已融入容器中，以支持广泛、易用的功能，例如应用性能管理 (APM)、云自动重新配置和容器安全性。最后但同样重要的是，Java 生成包配有标准 JRE，可直接使用或进行自定义设置。

Spring Boot 和 Java 生成包

如前所述，Java 生成包可为 Spring Boot 应用提供全面的容器自动化，这样您就可以推送打包 Spring Boot JAR 或 distZip 文件。是不是很方便？但它还有更多功能。Java 生成包可自动配置 Spring 云配置文件（如有）的注入。另外，它也可自动配置云级 Bean 和配置。

让我们总结一下

生成包在开发人员自由度和运维一致性之间达到了出色的平衡。它们可针对难以生成和管理的不同部分实现自动化，并通过抽象方式大大减少中间件的复杂性。现在它可真正提升 DevOps 的能力。

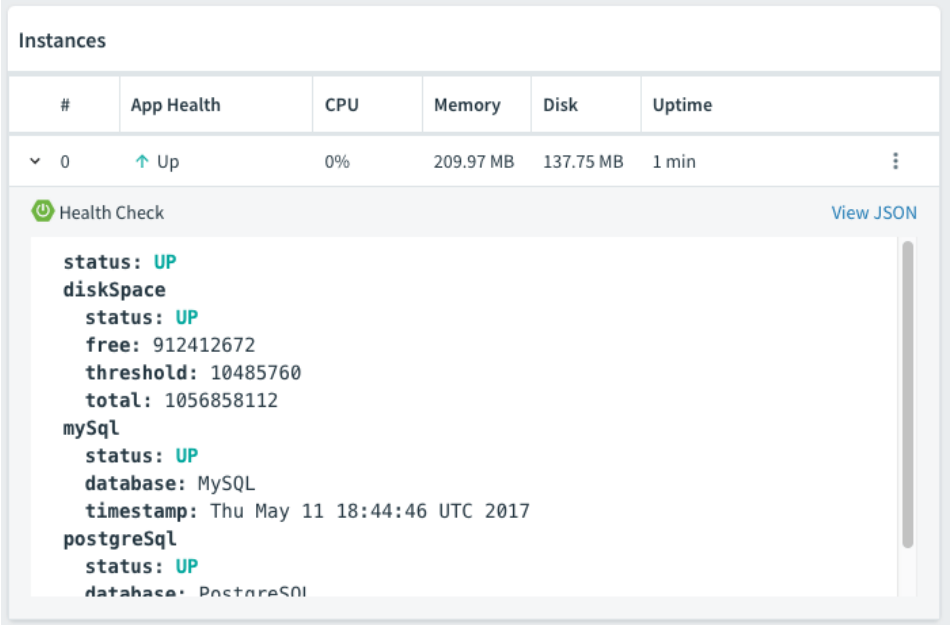
利用 Spring Boot Actuator 针对生产应用获得可见性

VMware 的支持者 Mark Heckler 写道：“应用在投入生产之前，都只是项目。投入生产后，开发人员和运维人员都需要针对关键应用获得最佳可见性，以跟踪出处，监控运行状况并快速针对意外行为的指标进行故障排除。”

Spring Boot Actuator 端点是获得此信息的有用方式。因此 Tanzu Application Service 在其应用管理器模块中集成了这些端点中的几个。
这些集成针对生产应用的状态提供有用的可见性。
以下是目前 TAS 支持的 Actuator 端点：

- /info - 公开有关应用环境、git 和版本的详细信息。
- /health - 通过安全连接展示运行状况或运行状况详细信息。
- /loggers - 列出应用中的 logger 级别，并允许修改。
- /dump - 生成线程转储。
- /trace - 显示您的应用最近 100 次 HTTP 请求的跟踪信息。
- /heapdump - 生成堆转储，并提供包含结果的压缩文件。
- /mappings - 显示应用提供服务的端点，以及其他相关详情。

一图道千言，以下屏幕截图可显示如何为您呈现此数据。



利用 Tanzu App Metrics 快速针对云原生应用进行故障排除

你是否还记得，我们说过，如果用平台来处理大部分复杂任务，微服务就会更加容易使用？Tanzu App Metrics 是 TAS 的集成指标模块，是此原则在实践中的出色应用案例。具体原因如下：

“如今的体系架构复杂性呈指数级增长。要提出的问题不是‘我的源代码出了什么问题？’而是需要在出现情况时由团队回答一系列问题：

- 工作负载的哪个组件出了问题？
- 如何在整个工作负载中跟踪相关请求？
- 如何从处理请求的组件中找到所有诊断信息？
- 我们如何尽快完成所有这些任务？

如果由不同团队负责工作负载的不同部分，这种情况会变得更加复杂。原因何在？只有少数人对整个工作负载有全面的理解。

但随着整个行业对微服务的依赖，我们也需要简化这些分布式现代系统的故障排除方式。”

App Metrics 是一款有用的工具，可帮助您大规模运行微服务。使用 App Metrics，您可以更快地回答这些问题：

- 哪里出了问题？出现问题的时间？应用指标可提供答案。这些指标可展示出现网络高延迟的时间段，以及平均响应时间。
- 为什么会出问题？日志和指标（例如 API 调用、具体响应时间和传递的参数）配合使用，通常可以提供足够的信息，找出出现延迟的原因。
- 谁是肇事者？事件包括外部信息，例如应用源代码何时变更，应用是否移动到性能低下的单元中的另一容器中。将事件与指标和日志关联通常是解开谜题的最后一步。

应用的遥测数据（日志、指标和事件）可以进行视觉渲染，还可提供交互式时间轴。因此更容易理解延迟来源和系统故障。App Metrics 还提供分布式跟踪功能 (Trace Explorer)，适用于启用了 Spring Coud Sleuth 的应用。这有助于您理解跨微服务调用方法的顺序。

对于 Spring Boot 开发人员来说，App Metrics 有一个特别的优势。还记得上一节的 Spring Boot Actuator 端点吗？App Metrics 可传入所有 Actuator 指标。此信息与已在 App Metrics 中直观显示的数据相结合，可为您提供更丰富的背景信息。您可以针对 Spring Boot 应用在生产中的运行状况获得更多见解。使用此功能缩短 Spring Boot 应用的问题平均解决时间 (MTTR)。

利用 App Autoscaler 毫不费力地扩展您的应用，满足客户需求您

的应用很有可能发生负载变化的情况。如何在不可预测的情况下扩展或缩减您的应用？您可以使用 App Autoscaler。此模块可实现应用横向扩展的自动化。出现流量高峰时为您的应用添加更多容量。流量恢复正常时进行缩减。最重要的是，您可以设置阈值和触发器，这样平台就可以根据您配置的规则为您进行扩展。

Loggregator 提供内置应用日志记录功能

如果您需要对应用进行故障排除，日志是必不可少的工具。因此 TAS 包括了 Loggregator，这个系统会收集日志和指标并流式传输。应用开发人员可以使用 Loggregator“跟踪”应用日志，或通过 Cloud Foundry Command Line Interface (cf CLI) 将最近的日志转储，或将它们流式传输到第三方日志归档和分析服务。

利用四层高可用性，让应用持续在线提供流量

TAS 通过四层堆栈提供冗余和恢复能力，即：监控进程、应用实例的运行状况管理、虚拟机 (VM) 的运行状况管理和可用区。

如果任何层中出现故障，TAS 会自动恢复，以持续提供流量。

需要缓存、消息代理或关系数据库？您可以使用服务代理。

您的应用要实现一些有意思的功能，就需要支持服务。Open Service Broker API 为您提供了一组标准命令和行为，它们在数百个不同的附加功能中保持一致。TAS 提供的所有服务均实施此 API。因此，您可以使用相同的重复活动创建并绑定丰富的服务。您是否需要缓存、消息代理、NoSQL 存储或关系数据库？您可以立即将它们添加到您的应用中。TAS 为 Redis、RabbitMQ 和 MySQL 提供即时可用的服务代理。这些代理可支持一系列服务器端应用场景，因此您可以专注于您的代码，在您需要的时候轻松添加您需要的内容。

感受到巨大威力了吗？这是因为不到一天时间想法就可转变为实际可用的软件。

总结一下。

在本白皮书的一开始，我们在寻求一些问题的答案，即“Tanzu Application Service 有何特别之处？”以及“为什么它可以如此出色地运行 Boot 应用？”那么现在，您已经了解，TAS 能够：

- 将应用上线必须完成的许多任务自动化
- 提供简单、一致、可配置的开发体验
- 生成容器自动化
- 提供一致的可配置操作模式

您还可以在 TAS 上原样运行 Spring Boot 应用，无需修改或只需很少修改。另外，TAS 在处理较大规模的 Spring 项目生态系统时表现良好，您可以随时轻松地为您的应用添加更多功能。

提到应用 - 您的企业中目前有多少个应用？几百个？几千个？无论有多少应用，TAS 都能圆满地完成任务。生成包和服务代理为您提供了很高的灵活性，可在平台上运行所有类型的应用。

使用 Spring Boot 和 TAS，您就踏上了超音速开发之路。当然，我们还有更长的路要走。请查看推荐阅读部分的链接，了解可帮助您更快一步的更多最佳实践！

推荐阅读

博客

- *Should that be a Microservice? Keep These Six Factors in Mind*
- *Getting Started with Spring Cloud Services on Tanzu Application Service*
- *Love Spring and Spring Boot? Then, You're Going to Love These Projects, Too*
- *Spring Boot 2.0 goes GA*
- *Opening Doors with Spring Boot 2.0*
- *How App Metrics Helps You Reduce MTTR for Spring Boot Apps (and Save Money too)*
- *TUTORIAL: Using Spring Boot Actuator Integrations With Tanzu Application Service*

幻灯片

- *Spring on TAS*

视频

- *Flight of the Flux: A Look at Reactor Execution Model*
- *Full Stack Reactive with React and Spring WebFlux*
- *Getting Super Productive with Spring Tools 4 and Spring Boot 2*
- *Guide to "Reactive" for Spring MVC Developers*
- *Scaling Spring Boot Applications in Real-Time*
- *Spring, the Kotlin and Functional Way*
- *Spring Boot 2.0 Web Applications*
- *Spring Cloud Stream – Developer Recipes, What's New and What's Next?*
- *Springing into Kotlin: How to Make the Magic Even More Magical*
- *Upgrading to Spring Boot 2.0*

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 vmware.com
威睿信息技术（中国）有限公司
北京朝阳区新源南路 8 号启皓北京东塔 8 层 801 邮编：100027 电话：+86-10-5976-6300
中国上海办公室 上海市淮海中路 333 号瑞安大厦 805B-809 室 邮编：200021 电话：+86-21-8024-9200
中国广州办公室 广州市天河路 385 号太古汇一座 3502 室 邮编：510610 电话：+86-20-87146110
中国香港公司 香港港岛东太古城太古湾道 12 号太古城中心 4 期 4 楼 电话：852-3696 6100 传真 852-3696 6101