Microsoft Azure 上部署 DevOps

2019 年 4 月

BESPIN GLOBAL

目录

前言

DevOps: Development 和 Operations 的组合，可以把 DevOps 看作开发（软件工程）、技术运营和质量保障（QA）三者的交集。
传统的软件组织将开发、IT 运营和质量保障设为各自分离的部门。在这种环境下如何采用新的开发方法（例如敏捷软件开发），这是一个重要的课题：按照从前的工作方式，开发和部署不需要 IT 支持或者 QA 深入的、跨部门的支持，而却需要极其紧密的多部门协作。然而 DevOps 考虑的还不止是软件部署。它是一套针对这几个部门间沟通与协作问题的流程和方法。
需要频繁交付的企业可能更需要对 DevOps 有一个大致的了解，DevOps 的引入能对产品交付、测试、功能开发和维护（包括——曾经罕见但如今已屡见不鲜的——"热补丁"）起到意义深远的影响。在缺乏 DevOps 能力的组织中，开发与运营之间存在着信息"鸿沟"——例如运营人员要求更好的可靠性和安全性，开发人员则希望基础设施响应更快，而业务用户的需求则是更快地将更多的特性发布给最终用户使用。这种信息鸿沟就是最常出问题的地方。
DevOps 汇集人员、流程和技术，实现软件交付自动化，为用户提供持续的价值。借助 Azure DevOps 解决方案，无论 IT 部门有多大规模或使用何种工具，都可以更加快速可靠地交付软件。

简化、加速并改善应用程序开发流程与体验

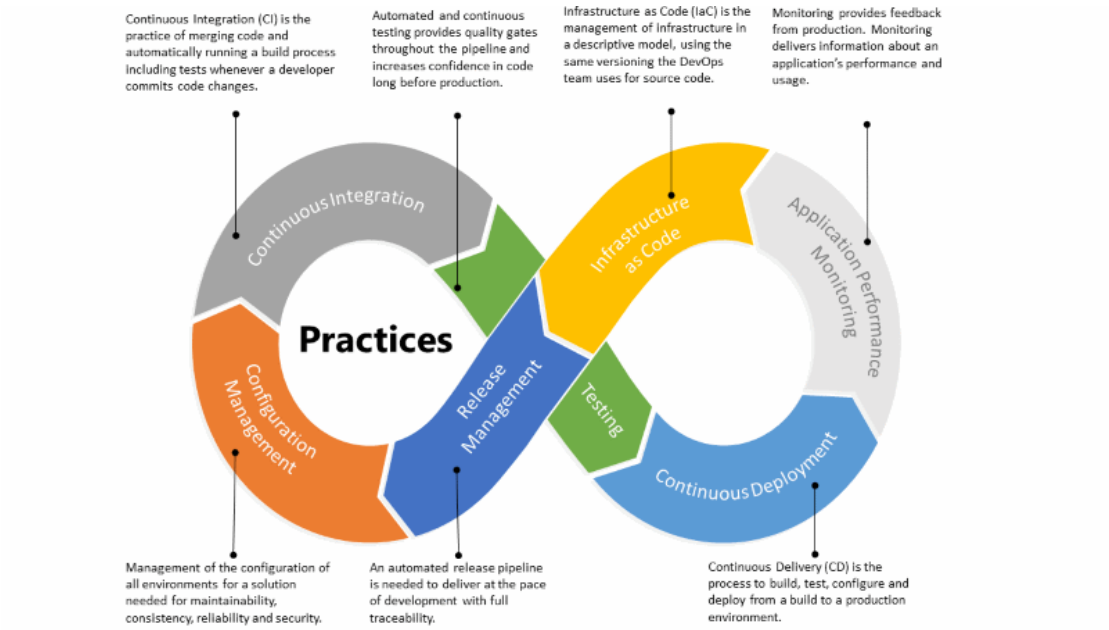
持续集成（CI）

利用持续集成改善软件开发质量、加快开发速度。如果使用 Azure DevOps 或 Jenkins 在云端构建应用并部署到 Azure，每次提交代码时都将自动进行构建和测试，从而更快检测到 bug。

持续交付（CD）

通过持续交付，确保代码和基础结构随时可以部署到生产环境。持续集成与基础结构即代码 (Infrastructure as Code) 的组合，不仅可以帮助你实现完全一致的部署，还让您随时都可以自信地在生产环境中进行手动部署。

CI/CD 管道是一系列的持续集成、 测试和部署到生产环境所做的更改所使用的非重复手段。



使用 CI/CD 实现持续部署

如果 CI/CD 测试成功，则可通过持续部署自动执行从代码提交到生产的整个过程。如果将 CI/CD 做法与监视工具配合使用，你将能够毫无延迟地将准备就绪的功能安全地提供给客户。

改善可靠性和可重复性

使用 IaC 自动预配和配置环境。以声明性代码的形式捕获环境定义，如 JSON 或 YAML。然后，使用 DevOps 工具（包括 Azure 资源管理器、Terraform 或 Ansible）以可靠的方式预配完全相同的环境。

利用日志分析工具获得可视化结果呈现

使用 Azure Log Analytics 和 Azure Monitor 监视基础结构运行状况，并集成至现有的仪表板，如 Grafana 或 Kibana。Azure Application Insights 通过应用程序性能管理和即时分析提供可操作见解。

加强合规性和安全性

你可以使用 Chef Automate 或 Azure Policy 等 DevOps 工具管理预配的基础结构和应用程序，从而确保符合性。在此基础上使用 Azure 安全中心等服务，可以更好地屏蔽风险，并快速找到和消除漏洞。

Azure DevOps Projects

使用 Azure DevOps 项目中，你可以构建 Azure 应用程序，在 Azure 服务，以分钟为单位。你还可以获取自动完全 CI/CD 管道集成、 内置监视和平台的所选的部署。中的表中所示图 2，有几种应用程序、 框架和部署类型时创建新的 DevOps 项目进行选择。请务必要注意此表反映在撰写本文时的信息并且由你阅读此内容的时间将改进了内置支持。

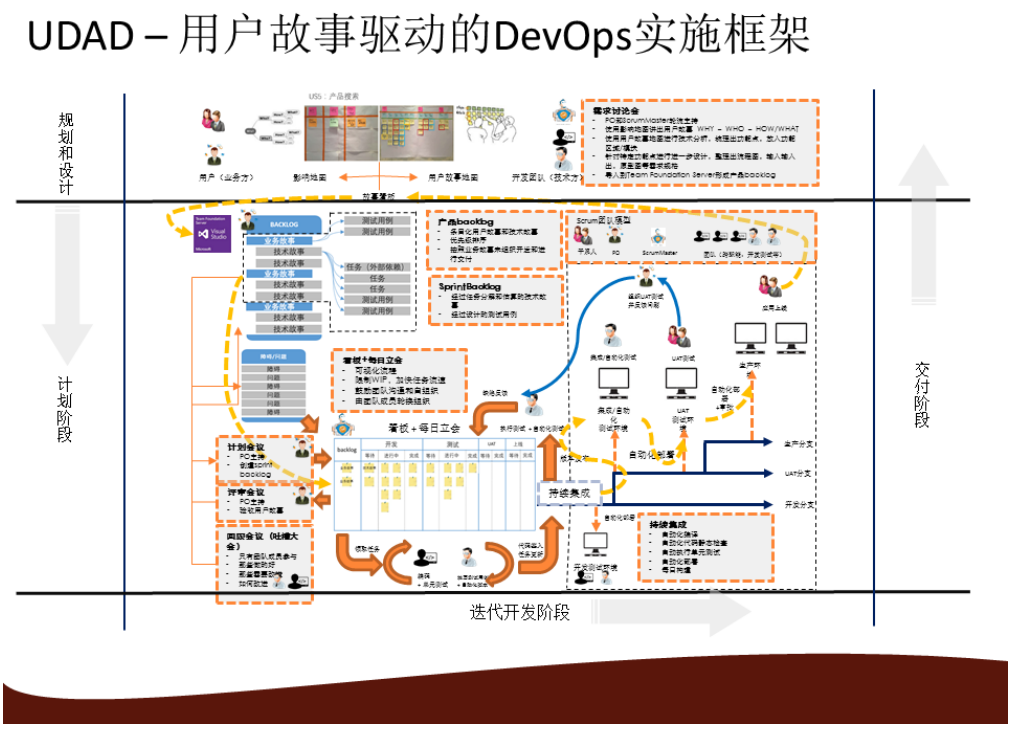
图 2 应用程序、 框架和部署选项

应用程序	框架	Windows 部署	Linux 部署

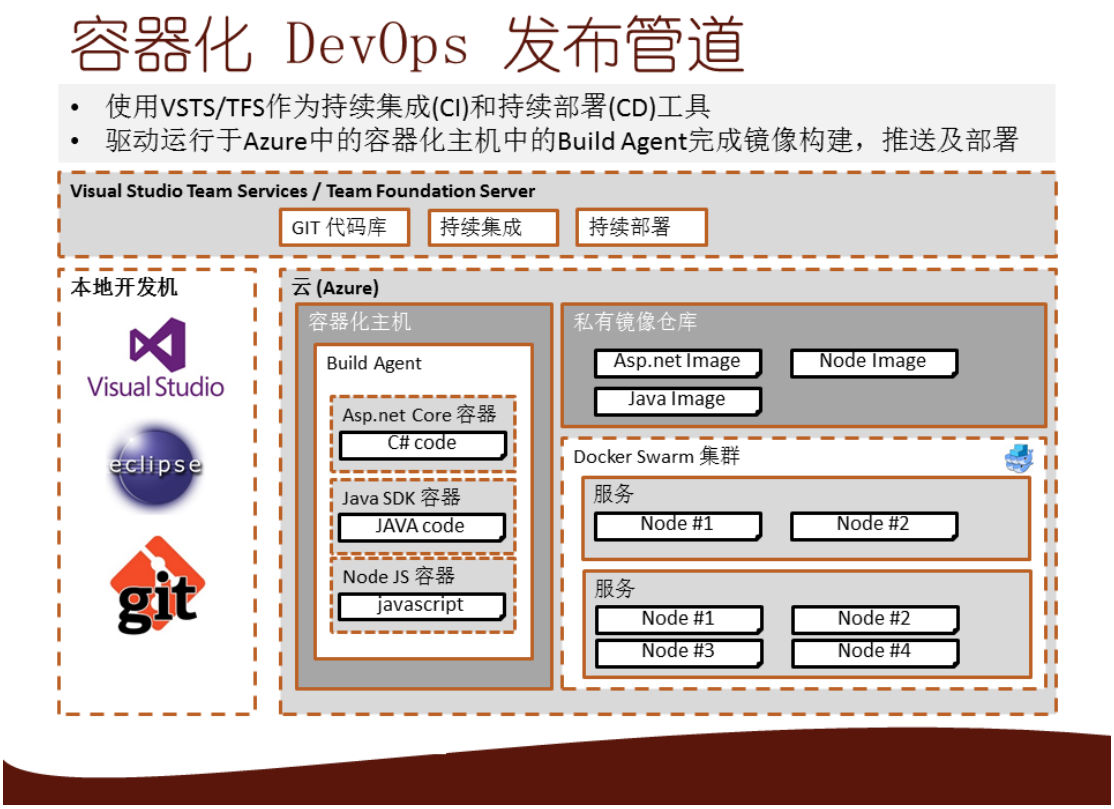
		Web 应用程序	虚拟机	Web 应用程序	容器的 web 应用
.NET	ASP.NET	X	X		
	ASP.NET Core	X	X	X	X
Java	Sprint	X			X
	JSF	X			X
Node.js	Express.js	X			X
	Sails.js				X
	Simple Node.js	X			X
PHP	Laravel	X			X
	Simple PHP	X			X
Python	Django	X		X	
	Bottle	X		X	
	Flask	X		X	X

基于 Azure 和 Team Foundation Server

软 DevOps 工具链是基于 Visual Studio 应用生命周期管理（VSALM - Visual Studio Application Lifecycle Management）的软件管理平台，本动手实验希望通过模拟一个产品从需求到上线的全过程让参与者感受到微软 DevOps 工具链端到端的管理能力。



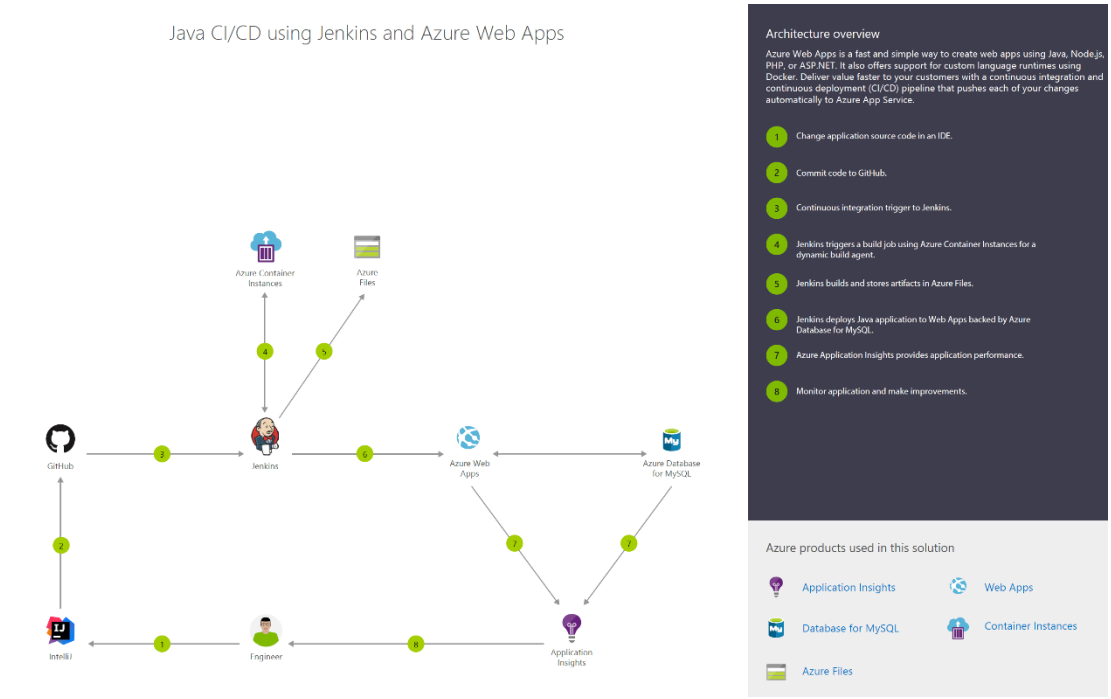
容器化 DevOps 发布管道



解决方案概述

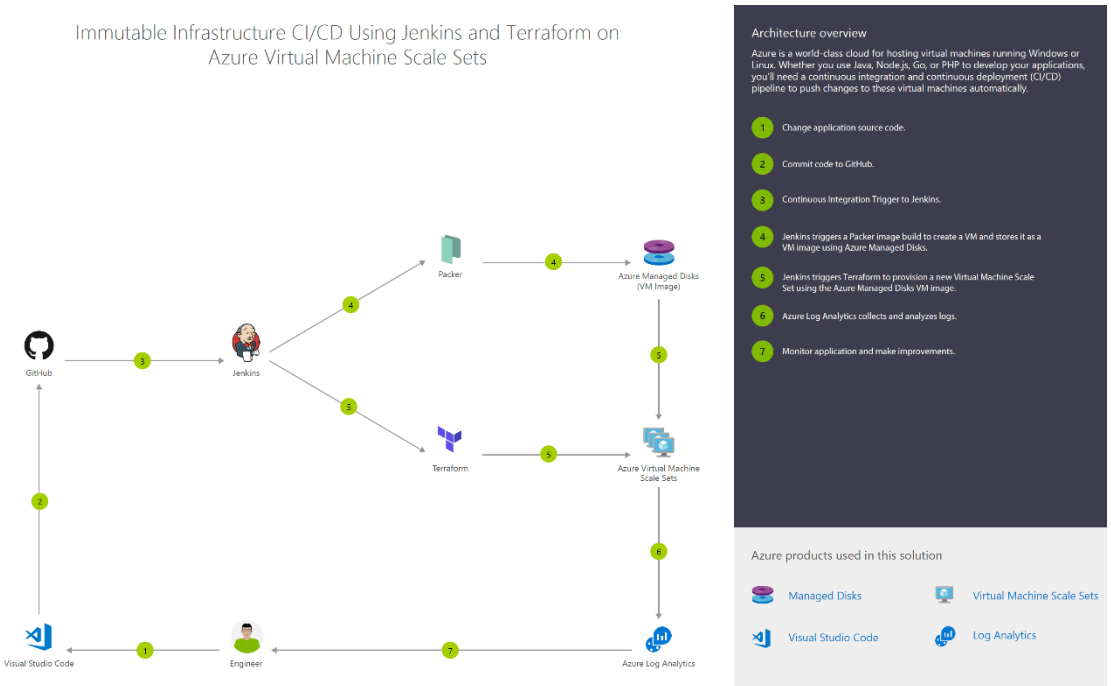
使用 Jenkins 和 Azure Web 的 Java CI/CD

借助 Azure 应用服务，你可以快速轻松地使用 Java、Node、PHP 或 ASP.NET 来创建 Web 应用，并使用 Docker 支持自定义语言运行时。持续集成和持续部署 (CI/CD) 管道可将每一个更改自动推送到 Azure 应用服务，让你更快地为客户创造价值。



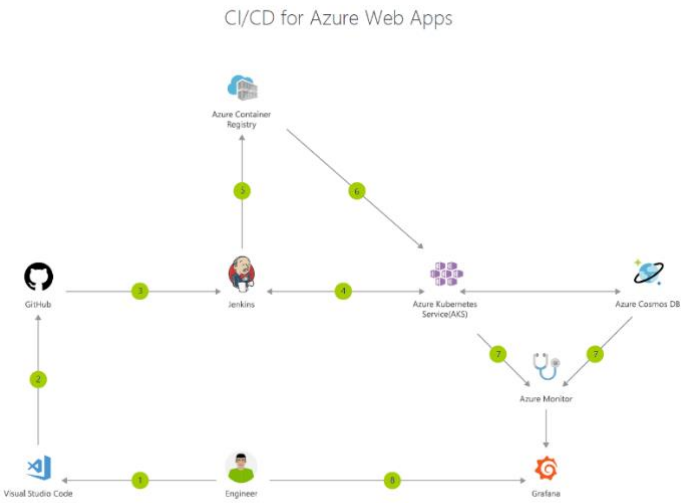
使用 Jenkins 和 Terraform 的 CI/CD

Azure 是用于托管运行 Windows 或 Linux 的虚拟机的世界级云。无论使用 Java、Node.js 还是 PHP 来开发应用程序，都需要持续集成和持续部署（CI/CD）管道自动将更改推送到这些虚拟机。



使用 Jenkins 和 Kubernetes 的容器 CI/CD

使用容器，可轻松地持续生成和部署应用程序。使用 Azure Kubernetes 服务 (AKS) 协调这些容器的部署，获得可复制、可管理的容器群集。
通过设置持续生成来生成容器映像和业务流程，可更快更可靠地进行部署。



示例

使用 Jenkins 和 Azure DevOps Services 将应用部署到 Azure 中的 Linux 虚拟机

持续集成 (CI) 和持续部署 (CD) 可形成一个管道，用于生成、发布和部署代码。 Azure DevOps Services 针对到 Azure 的部署提供了一组完整的功能完备的 CI/CD 自动化工具。 Jenkins 是一款常用的基于服务器的第三方 CI/CD 工具，也提供 CI/CD 自动化。 可以结合使用 Azure DevOps Services 和 Jenkins 来自定义交付云应用或服务的方式。

在本例中，将使用 Jenkins 生成 Node.js Web 应用。 然后使用 Azure DevOps 将该应用部署到

包含 Linux 虚拟机 (VM) 的部署组。 需要执行以下操作：

- 获取示例应用。
- 配置 Jenkins 插件。
- 为 Node.js 配置 Jenkins 自由风格项目。
- 为 Azure DevOps Services 集成配置 Jenkins。
- 创建 Jenkins 服务终结点。
- 为 Azure 虚拟机创建部署组。
- 创建 Azure Pipelines 发布管道。

- 执行手动和 CI 触发的部署。

准备阶段

- 需要具有对 Jenkins 服务器的访问权限。 如果尚未创建 Jenkins 服务器，请参阅 Jenkins 文档。
- 登录到 Azure DevOps Services 组织 (**https://{yourorganization}.visualstudio.com**)。 可以免费创建 Azure DevOps Services 组织。

  备注

  有关详细信息，请参阅**连接到 Azure DevOps Services**。

- 部署目标需用到 Linux 虚拟机。 有关详细信息，请参阅使用 Azure CLI 创建和管理 Linux VM。
- 为虚拟机开启入站端口 80。 有关详细信息，请参阅使用 Azure 门户创建网络安全组。

获取示例应用

你需要一个待部署的应用，并存储在 Git 存储库中。 对于本教程，我们建议你使用从 GitHub 获得的此示例应用。 本教程包含用于安装 Node.js 和应用程序的示例脚本。 要使用自己的存储库，请配置一个类似的示例。

创建此应用的一个分支并记下位置 (URL) 以便在本教程的后续步骤中使用。 有关详细信息，请参阅分支存储库

备注

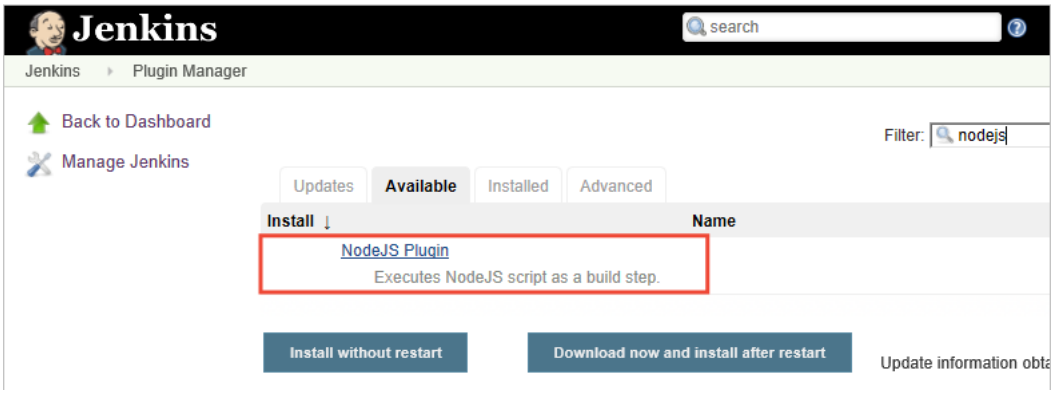
应用是通过 **Yeoman** 生成的。 它使用 Express、bower 和 grunt。 它还有一些 npm 包作为依赖项。 示例还包含一个用来设置 Nginx 并部署应用的脚本。 它在虚拟机上执行。 具体而言，脚本将执行以下操作：

1. 安装 Node、Nginx 和 PM2。
2. 配置 Nginx 和 PM2。
3. 启动 Node 应用。

配置 Jenkins 插件

首先，必须配置两个 Jenkins 插件：NodeJS 和 VS Team Services 持续部署。

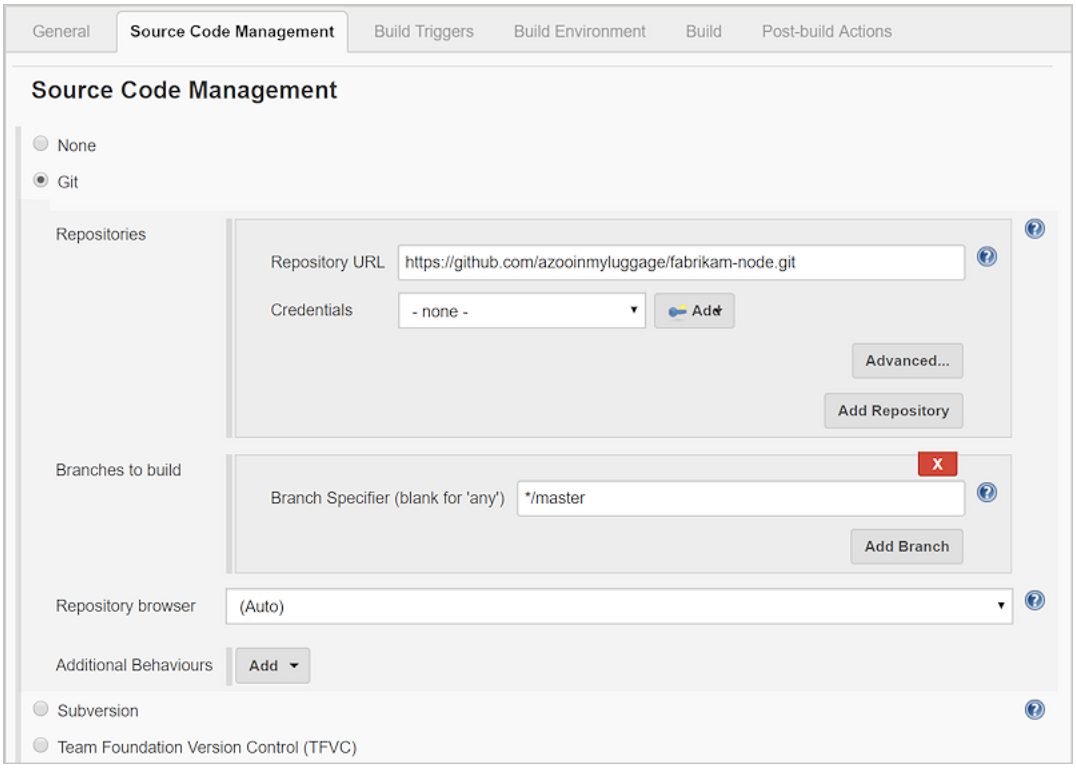
1. 打开你的 Jenkins 帐户，并选择“管理 Jenkins”。
2. 在“管理 Jenkins”页面上，选择“管理插件”。
3. 筛选列表以查找“NodeJS”插件，然后选择“安装而不重启”选项。



1. 筛选列表以查找“VS Team Services 持续部署”插件，然后选择“安装而不重启”选项。
2. 返回 Jenkins 仪表板，然后选择“管理 Jenkins”。
3. 选择“全局工具配置”。 查找“NodeJS”，然后选择“NodeJS 安装”。
4. 选择“自动安装”选项，然后输入“名称”值。
5. 选择“其他安全性验证” 。

为 Node.js 配置 Jenkins 自由风格项目

1. 选择“新建项”。 输入项目值。
2. 选择“自由风格项目”。 选择“确定” 。
3. 在“源代码管理”选项卡上，选择“Git”并输入包含应用代码的存储库和分支的详细信息。



为 Azure DevOps Services 集成配置 Jenkins

备注

确保用于以下步骤的个人访问令牌（PAT）包含 Azure DevOps Services 中的“发布”（读取、写入、执行和管理）权限。

1. 如果没有 PAT，请在 Azure DevOps Services 组织中创建一个 PAT。 Jenkins 需要使用此信息来访问你的 Azure DevOps Services 组织。 确保存储令牌信息以用于本部分后面的步骤。

   若要了解如何生成令牌，请阅读如何为 Azure DevOps Services 创建个人访问令牌？。

2. 在“生成后操作”选项卡中，选择“添加生成后操作”。 选择“存档项目”。
3. 对于“要存档的文件”，输入 **/* 以包括所有文件。
4. 若要创建其他操作，请选择“添加生成后操作”。
5. 选择“在 TFS/Team Services 中触发发布”。 输入 Azure DevOps Services 组织的 URI，例如 **https://{your-organization-name}.visualstudio.com**。
6. 输入**项目**名称。
7. 为发布管道选择名称。 （稍后将在 Azure DevOps Services 中创建此发布管道。）
8. 选择用于连接到 Azure DevOps Services 或 Team Foundation Server 环境的凭据：
   - 如果使用的是 Azure DevOps Services，请将“用户名”留空。

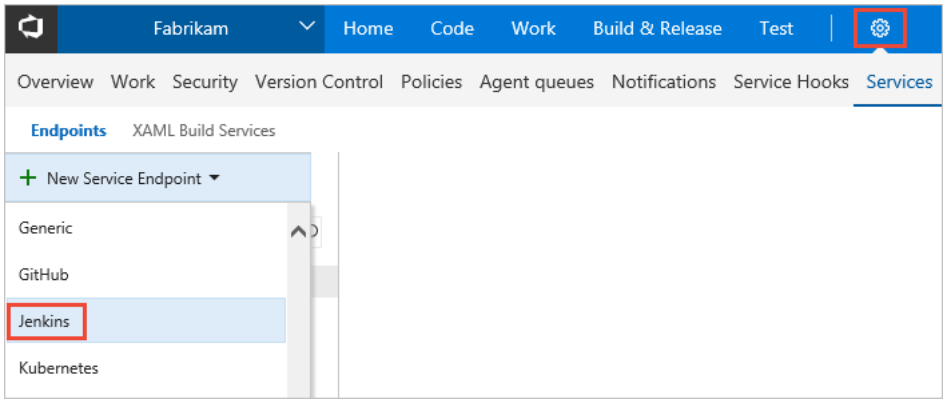
- 如果你正在使用 Team Foundation Server 的本地版本，请输入用户名和密码。

9. 保存 Jenkins 项目。

创建 Jenkins 服务终结点

服务终结点允许 Azure DevOps Services 连接到 Jenkins。

1. 在 Azure DevOps Services 中打开“服务”页面，打开“新服务终结点”列表，然后选择“Jenkins”。



2. 输入连接的名称。
3. 输入 Jenkins 服务器的 URL，然后选择“接受不受信任的 SSL 证书”选项。 示例 URL 是 http://{YourJenkinsURL}.chinaeast.cloudapp.chinacloudapi.cn。
4. 输入 Jenkins 帐户的用户名和密码。
5. 选择“验证连接”确认信息是否正确。
6. 选择“确定”，创建服务终结点。

为 Azure 虚拟机创建部署组。

需要使用部署组来注册 Azure DevOps Services 代理，以便可将发布管道部署到虚拟机。 通过部署组，可以方便地定义目标计算机的逻辑组，以进行部署，并在每台计算机上安装所需的代理。

备注

在以下过程中，确保安装先决条件，且不使用 sudo 权限运行脚本。

1. 打开“生成 & 发布”中心的“发布”选项卡，打开“部署组”，然后选择“+ 新建”。
2. 为部署组输入名称和可选说明。 然后选择“创建”。
3. 为部署目标虚拟机选择操作系统。 例如，选择“Ubuntu 16.04+”。
4. 选择“使用脚本中的个人访问令牌进行身份验证”。
5. 选择“系统必备组件”链接。 为操作系统安装必备组件。

6. 选择“将脚本复制到剪贴板”以复制脚本。
7. 登录到部署目标虚拟机并运行脚本。 不要使用 sudo 权限运行脚本。
8. 完成安装后，系统将提示你部署组标记。 接受默认值。
9. 在 Azure DevOps Services 中，在“部署组”下面的“目标”中检查新注册的虚拟机。

创建 Azure Pipelines 发布管道

发布管道指定 Azure Pipelines 在部署应用时所用的过程。 在此示例中，你将执行一个 shell 脚本。

在 Azure Pipelines 中创建发布管道：

1. 打开“生成和发布”中心的“发布”选项卡，然后选择“创建发布管道”。
2. 通过选择以“空进程”开始，选择“空”模板。
3. 在“项目”部分，选择“+ 添加项目”，然后在“源类型”中选择“Jenkins”。 选择自己的 Jenkins 服务终结点连接。 然后选择 Jenkins 源作业，并选择“添加”。
4. 选择“环境 1”旁边的省略号。 选择“添加部署组阶段”。
5. 选择部署组。
6. 选择“+”，向“部署组阶段”中添加任务。
7. 选择“Shell 脚本”任务，并选择“添加”。 “Shell 脚本”任务为要在每个服务器上运行的脚本提供配置，以安装 Node.js 并启动应用。
8. 对于“脚本路径”，请输入“$(System.DefaultWorkingDirectory)/Fabrikam-Node/deployscript.sh”。
9. 选择“高级”，然后启用“指定工作目录”。
10. 对于“工作目录”，请输入“$(System.DefaultWorkingDirectory)/Fabrikam-Node”。
11. 将发布管道的名称编辑为你在 Jenkins 内部版本的“生成后操作”选项卡上指定的名称。 Jenkins 要求此名称能够在源项目更新时触发新的发布。
12. 选择“保存”，然后选择“确定”以保存发布管道。

执行手动和 CI 触发的部署

1. 选择“+ 发布”，然后选择“创建发布”。
2. 选择在突出显示的下拉列表中完成的内部版本，然后选择“队列”。
3. 在弹出消息中选择发布链接。 例如：“发布 **Release-1** 已创建。”
4. 打开“日志”选项卡以查看发布控制台输出。
5. 在浏览器中，打开已添加到部署组的其中一个服务器的 URL。 例如，输入 http://{your-server-ip-address}。
6. 转到源 Git 存储库，使用某些已更改的文本来修改文件 app/views/index.jade 中 h1 标题的内容。
7. 提交更改。
8. 几分钟后，可在 Azure DevOps 的“发布”页中看到新建的发布管道。 打开此发布可以看到部署正在进行。

排查 Jenkins 插件问题

如果 Jenkins 插件出现任何 bug，请在 Jenkins JIRA 中提出特定组件的问题。

后续步骤

在本示例中，我们已使用 Jenkins 进行生成并使用 Azure DevOps Services 进行发布，自动将一个应用部署到 Azure。 你已了解如何：

- 在 Jenkins 中生成应用。
- 为 Azure DevOps Services 集成配置 Jenkins。
- 为 Azure 虚拟机创建部署组。
- 创建用于配置 VM 和部署应用的发布管道。