
欢迎学习模块 10：实现架构自动化。


本模块包含以下章节：

1. 架构需求
2. 实现自动化的原因
3. 实现基础设施自动化
4. 实现部署自动化
5. AWS Elastic Beanstalk

本模块还包括：

- 讲师指导的演示，*首先分析 AWS CloudFormation 模板的结构，然后根据模板创建堆栈。*
- 指导实验，*提供使用 AWS CloudFormation 在 AWS 账户中创建资源的实践机会。*
- 挑战实验，*使用 AWS CloudFormation 创建支持咖啡馆使用案例的 Amazon Web Services (AWS) 资源。*

最后，您需要完成一个*知识测验*，以测试您对本模块中涵盖的关键概念的理解程度。

# 模块目标

学完本模块后，您应该能够：

- 识别何时实现自动化及原因
- 确定如何使用 AWS CloudFormation 建模、创建和管理 AWS 资源集合
- 使用 Quick Start AWS CloudFormation 模板设置架构
- 说明如何借助 AWS System Manager 和 AWS OpsWorks 实现基础设施和部署自动化
- 说明如何使用 AWS Elastic Beanstalk 部署简单的应用程序

学完本模块后，您应该能够：

- 识别何时实现自动化及原因
- 确定如何使用 AWS CloudFormation 建模、创建和管理 AWS 资源集合
- 使用 Quick Start AWS CloudFormation 模板设置架构
- 说明如何借助 AWS System Manager 和 AWS OpsWorks 实现基础设施和部署自动化
- 说明如何使用 AWS Elastic Beanstalk 部署简单的应用程序

模块 10：实现架构自动化

# 第 1 节：架构需求

介绍第 1 节：架构需求。

# 咖啡馆业务要求

咖啡馆现在在多个国家/地区设有分店，必须开始自动化才能保持发展。其组织有许多不同的架构，需要一种方法对它们进行统一的部署、管理和更新。

到目前为止，咖啡馆创建了 AWS 资源并手动配置了应用程序 – 主要使用 AWS 管理控制台。这种方法可以有效帮助咖啡馆快速开发 Web 服务，构建支持员工和客户需求的基础设施。但是，他们发现将部署复制到新的 AWS 区域，以支持在多个国家/地区设立的咖啡馆分店挑战重重。

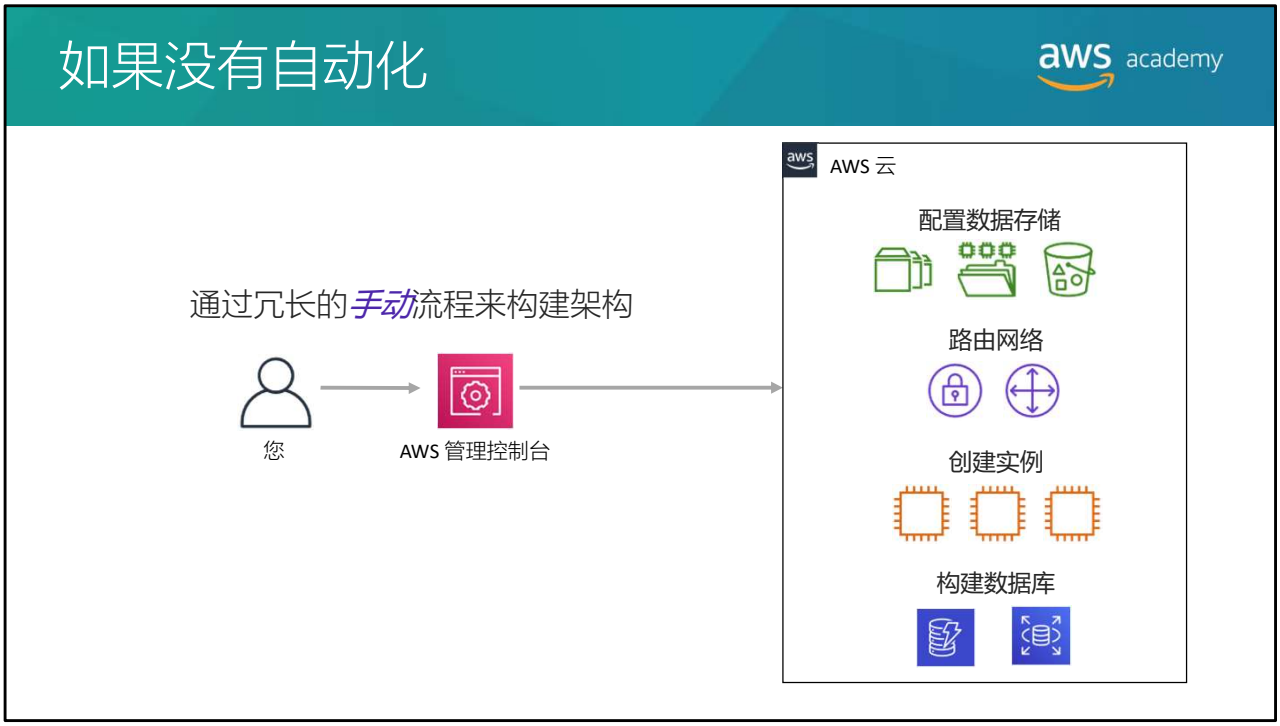
他们还希望拥有具有可靠匹配配置的独立开发和生产环境。他们意识到，必须开始自动化才能支持持续增发展。其组织有许多不同的架构，需要一种方法快速、一致且可靠地部署、管理和更新这些架构。

在本模块中，您将了解提供自动化的 AWS 服务，包括 AWS CloudFormation。通过使用 AWS CloudFormation，您将能够帮助咖啡馆满足这些新的业务需求。

模块 10：实现架构自动化

# 第 2 节：实现自动化的原因

介绍第 2 节：实现自动化的原因。



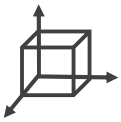

构建大规模计算环境需要耗费大量的时间和精力。

许多组织开始使用 AWS 的方法是，手动创建 Amazon Simple Storage Service (Amazon S3) 存储桶或启动 Amazon Elastic Compute Cloud (Amazon EC2) 实例并在其上运行 Web 服务器。然后，随着时间的推移，他们会手动添加更多资源，因为他们发现扩大对 AWS 的使用可以满足其他业务需求。但是，手动管理和维护这些资源很快就会变得具有挑战性。

要问的一些问题包括：

- 您要将主要精力用在何处：设计还是实施？手动实施有哪些风险？
- 理想情况下，如何更新生产服务器？如何在多个地理区域推出部署？当出现问题时，又如何回滚到上个已知良好版本？
- 如何调试部署？在向客户推出部署之前，能否修复应用程序中的错误？如何发现问题所在，然后修复问题，以使其*保持*修复状态？
- 如何管理对组织中各种系统和子系统的依赖关系？
- 最后，通过手动配置完成所有这些任务是否现实？

## 手动流程带来的风险



### 不支持大规模可重复性

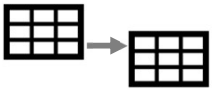
- 如何将部署复制到多个区域？

### 没有版本控制

- 如何将生产环境回滚到以前的版本？

### 缺少审计跟踪

- 如何确保合规性？如何在资源级别跟踪对配置详细信息的更改？



### 数据管理不一致

- 例如，如何确保跨多个 Amazon Elastic Compute Cloud (Amazon EC2) 实例的配置匹配？

手动创建资源并向环境添加新的特性和功能*无法扩展*。如果您负责的是大型企业应用程序，则会面临人手不足的困境。

此外，从头开始创建架构和应用程序没有固有的*版本控制*。在紧急情况下，将生产堆栈回滚到之前的版本很有用，但如果是手动创建环境，就不可能这样做。

对于许多合规性和安全性相关情况，拥有*审计跟踪*非常重要。允许组织中的任何人手动控制和编辑环境存在风险。

最后，要最大限度地降低风险，*一致性*至关重要。自动化使您能够保持一致性。

## 遵循 AWS 架构完善框架的原则

- 卓越运营设计原则
  - 以代码形式运营
  - 进行频繁、可逆的微小更改
- 可靠性支柱设计原则
  - 管理自动化方面的变更

按照手动方法创建和维护 AWS 资源和部署并不能使您满足这些准则。

还要考虑手动方法符合 AWS 架构完善的框架的程度。卓越运营的六项设计原则之一是*以代码形式运营*。在云中，您可以将用于应用程序代码的相同的工程学科应用于整个环境。您可以将整个工作负载（应用程序、基础设施和其他资源）定义为代码，并使用代码进行更新。您可以编写运行程序脚本，并在运行时通过触发它们来响应事件以实现自动化。通过以代码形式运营，您可以限制人为错误并实现对事件的一致响应。

另一项卓越运营设计原则是*进行频繁、可逆的微小更改*。也就是说，设计工作负载以支持对组件的定期更新，以便您增加工作负载中的有益变化。以较小的增量进行更改，如果这些更改无法帮助您识别和解决环境中引入的问题，则可以撤消这些更改。如果可能，在进行这些更改时尽量不要影响客户。

最后，架构完善的可靠性支柱 的设计原则之一是*管理自动化更改*。对基础设施的更改应通过自动化来完成。因此，必须管理对自动化的更改。 对许多组织来说，更改生产系统是面临的最大风险领域之一。在操作中，尽可能使用自动化，比如测试和部署更改、添加或删除容量以及迁移数据。

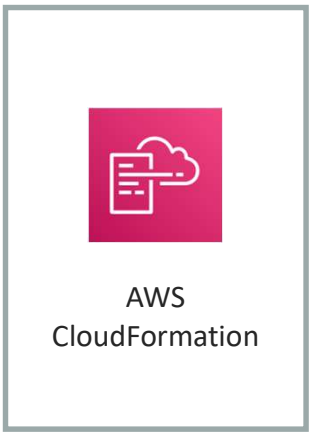
模块 10：实现架构自动化

# 第 3 节：实现基础设施自动化

aws academy

介绍第 3 节：实现基础设施自动化。

# 实现基础设施自动化

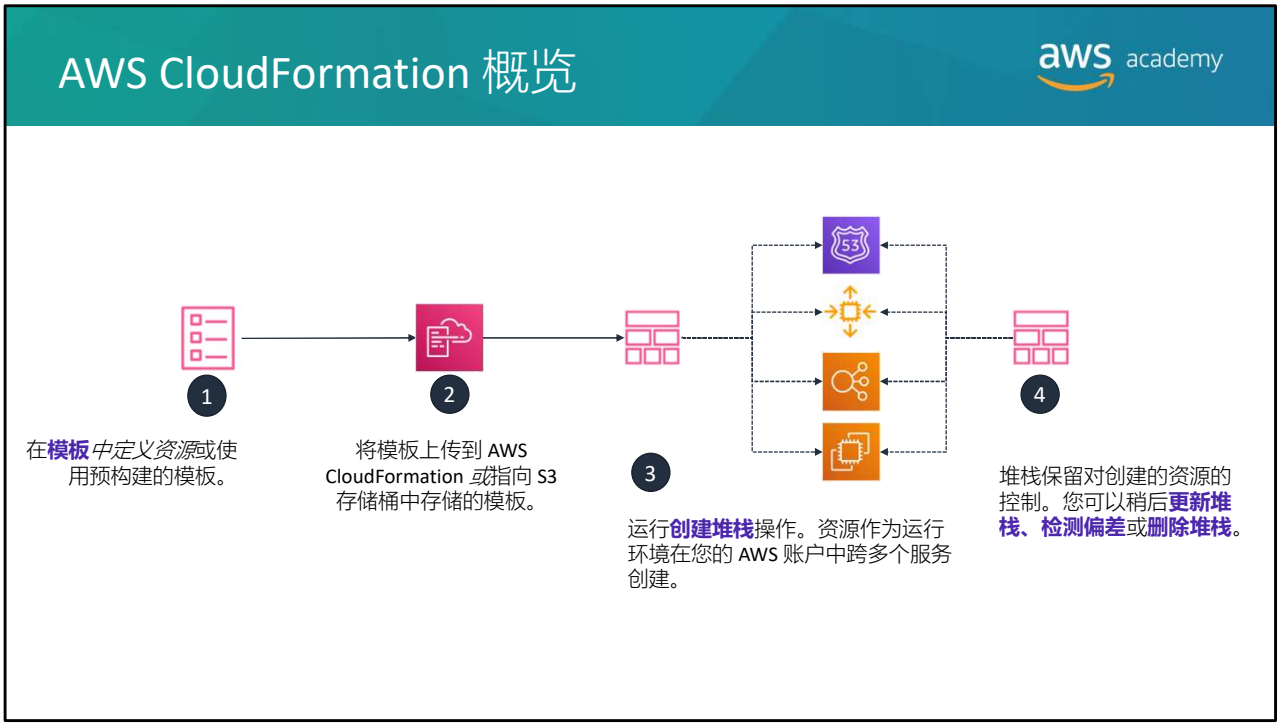



- AWS CloudFormation 提供了一种简化的方法来建模、创建和管理 AWS 资源的集合
  - 资源集合称为 AWS CloudFormation 堆栈
  - 无需支付额外费用（仅为您创建的资源付费）
- 可以创建、更新和删除堆栈
- 支持有序且可预测的资源预置和更新
- 启用 AWS 资源部署的版本控制

AWS CloudFormation 以可重复的方式预置资源。它允许您构建和重建基础设施和应用程序，无需执行手动操作或编写自定义脚本。使用 AWS CloudFormation，您可以编写一个文档来描述您的基础设施应该是什么，包括应作为部署一部分的所有 AWS 资源。您可以将这份文档视为*模型*。然后，使用该模型创建资源，因为 AWS CloudFormation 实际上可以在您的账户中创建资源。

当您使用 AWS CloudFormation 创建资源时，这称为 AWS CloudFormation *堆栈*。您可以创建堆栈、更新堆栈或删除堆栈。因此，您可以按有序且可预测的方式预置资源。

使用 AWS CloudFormation，您可以将基础设施视为代码 (IaC)。您可以使用任意代码编辑器进行编写，将其签入*版本控制*系统（如 GitHub 或 AWS CodeCommit）中，并在部署到相应环境之前与团队成员一起审核文件。如果您创建用于对部署建模的 AWS CloudFormation 文档已签入版本控制系统，则您始终可以删除堆栈，签出旧版本的文档，然后从中创建堆栈。通过版本控制，您可以使用基本的回滚功能。




该图演示了 AWS CloudFormation 的工作原理。首先，定义要创建的 AWS 资源。在此示例中，创建了几个 EC2 实例、一个负载均衡器、一个 Auto Scaling 组和一个 Amazon Route 53 托管区域。在 AWS CloudFormation *模板*中定义资源。您可以从头开始创建模板，也可以使用预构建的模板。许多示例模板也可供使用。

尽管 AWS CloudFormation 为 AWS 服务提供了广泛的支持，但并非所有资源都可以由 AWS CloudFormation 创建。有关详细信息，请参阅 AWS CloudFormation 支持的资源列表。

接下来，将模板上传到 AWS CloudFormation。或者，您可以将模板存储在 Amazon S3 上，然后将 AWS CloudFormation 指向存储它的位置。

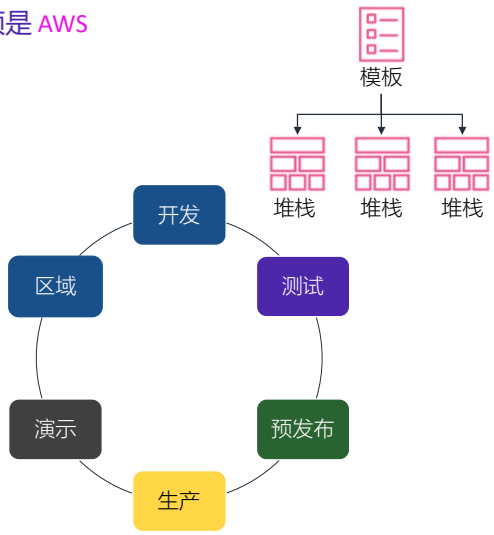
第三，运行*创建堆栈*操作。执行此操作时，AWS CloudFormation 服务将读取模板中指定的内容，并在您的 AWS 账户中创建所需的资源。单个堆栈可以跨多个 AWS 服务在单个区域中创建和配置资源。

最后，您可以观察堆栈创建过程的进度。堆栈成功完成后，它创建的 AWS 资源就存在于您的账户中。堆栈对象仍然存在，其作用就像创建的所有资源的句柄一样。当您以后要执行操作时，这很有用。例如，您可能想要*更新堆栈*（创建其他 AWS 资源或修改现有资源）或*删除堆栈*（清理和删除堆栈创建的资源）。

# 基础设施即代码 (IaC)

对于 AWS 云开发，IaC 的内置选项是 AWS CloudFormation。

- IaC 是通过编写具有以下特性的模板文件来预置和管理您的云资源的过程 –
  - 人类可读
  - 机器可用
- 它是可以复制、重新部署和重新定位的基础设施
- 您可以在出现故障时回滚到上一个良好状态

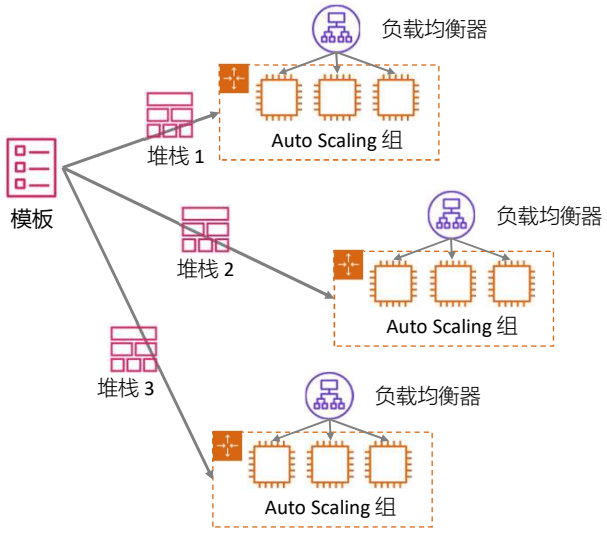



基础设施即代码 (IaC) 是一个行业术语，它指的是通过在人类可读和机器可用的模板文件中定义云资源来进行预置和管理的过程。

IaC 越来越受欢迎，因为它提供了一个可行的解决方案来应对各种挑战，例如如何轻松、可靠且一致地复制和重新部署基础设施以及重新调整其用途。

从客户角度来看，AWS CloudFormation 的事务性是其最大的优势之一。AWS CloudFormation 是事务性的 – 在出现故障时服务将*回滚到*上一个良好状态。

# 基础设施即代码：优势




## 减少多个匹配环境

- 快速部署复杂环境
- 提供配置一致性
- 需要时可轻松清理（删除堆栈会删除创建的资源）
- 轻松将更改传播到所有堆栈
  - 修改模板，在所有堆栈上运行更新堆栈

## 优势

- 可重用性
- 可重复性
- 可维护性

现在，请详细考虑 IaC 的一些优势。如果您使用代码构建基础设施，将获得快速部署复杂环境的能力之类的好处。使用一个模板（或多个模板的组合），您可以重复构建相同的复杂环境。

在此示例中，一个模板可以用来创建三个不同的堆栈。每个堆栈通常可在几分钟内快速创建。每个堆栈一致地复制复杂的配置详细信息。

如果堆栈 2 是您的测试环境，堆栈 3 是您的生产环境，您可以更有把握地认为，如果测试作业在测试环境中表现良好，那么它们也将在生产环境中表现良好。该模板将测试环境与生产环境配置不同的风险降至最低。

此外，如果必须在测试环境中进行配置更新，则可以通过更改更新模板并更新所有堆栈。该过程有助于确保对单一环境的修改可靠地传播到应接收更新的所有环境。

另一个好处是，当不再需要时，可以更轻松地*清理*在账户中创建的所有资源以支持测试环境。这有助于降低与您不再需要的资源相关的成本，并使您的账户保持简单。

# AWS CloudFormation 模板语法

aws academy

AWS CloudFormation 模板

- 使用 JavaScript Object Notation (JSON) 或 YAML Ain't Markup Language (YAML) 编写
- YAML 的优势 –
  - 不那么冗长（没有 {}、"" 字符）
  - 支持嵌入式备注
- JSON 的优势 –
  - 在其他计算机系统（例如 API）中得到了更广泛的应用
- 推荐 – 将模板视为源代码
  - 将它们存储在代码存储库中

```
{
 "AWSTemplateFormatVersion": "2010-09-09",
 "Resources" : {
 "awsexamplebucket1" : {
 "Type" : "AWS::S3::Bucket"
 }
 }
}
```

JSON 示例

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
 awsexamplebucket1:
 Type: AWS::S3::Bucket
```

YAML 示例

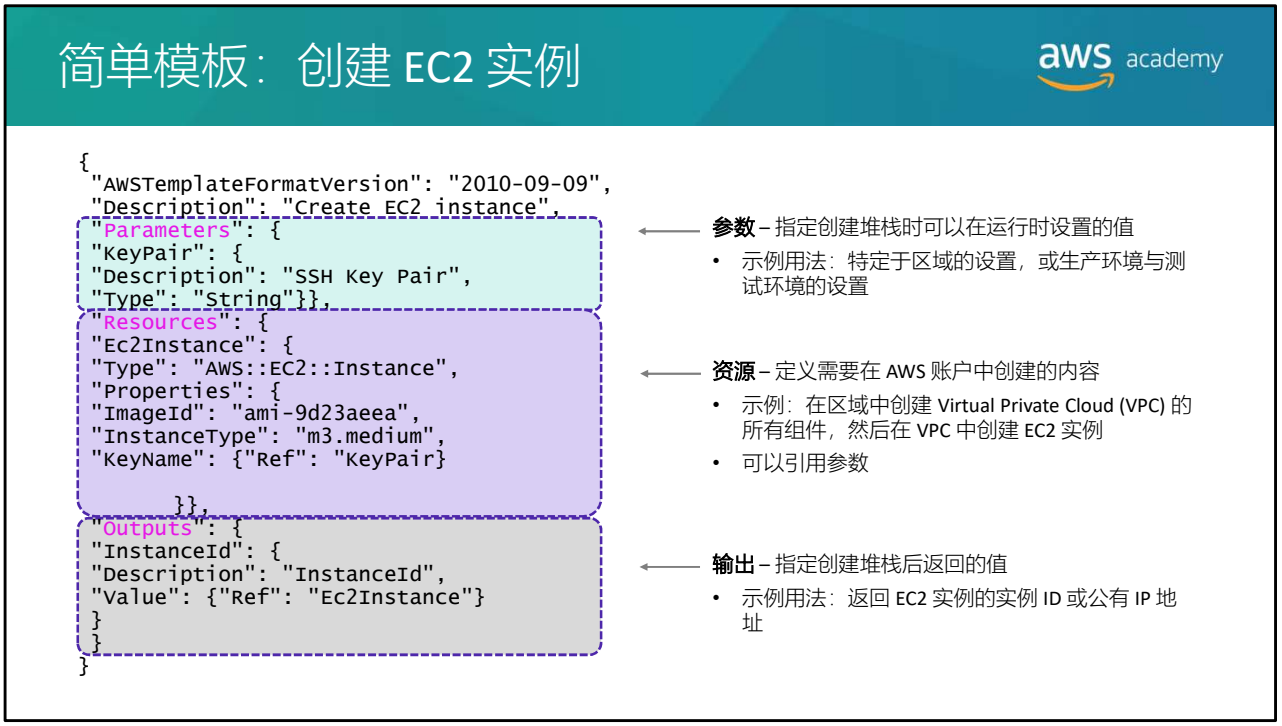
模板也可以在 AWS CloudFormation Designer（AWS 管理控制台中的图形设计界面）中创建。

AWS CloudFormation 模板可以使用 JavaScript Object Notation (JSON) 或 YAML Ain't Markup Language (YAML) 编写。

*YAML* 在可读性方面进行了优化。同样的数据以 YAML 格式存储要比以 JSON 格式存储所用的行数更少，因为 YAML 并不使用大括号 ({})，而且使用的引号 ( “” ) 也更少。YAML 的另一个优点是，它原生支持嵌入式备注。与 JSON 相比，调试 YAML 文档也更加轻松。使用 JSON 时，可能很难找到丢失或放错位置的逗号或大括号。而 YAML 不存在这个问题。

尽管 YAML 拥有很多优点，但 *JSON* 也具有一些独特的优点。首先，它在计算机系统中得到了广泛应用。它的普及率是一个优势，因为以 JSON 格式存储的数据能够可靠地与很多系统结合使用，无需进行转换。此外，生成和解析 JSON 通常要比生成生成和解析 YAML 更轻松。

AWS 管理控制台提供了一个名为 *AWS* CloudFormation Designer 的图形界面，可用于编写或查看 AWS CloudFormation 模板的内容。它还可以用来将有效的 JSON 模板转换为 YAML 或将 YAML 转换为 JSON。它提供了一个用来编写模板的拖放界面，这些模板可以 JSON 或 YAML 格式输出。

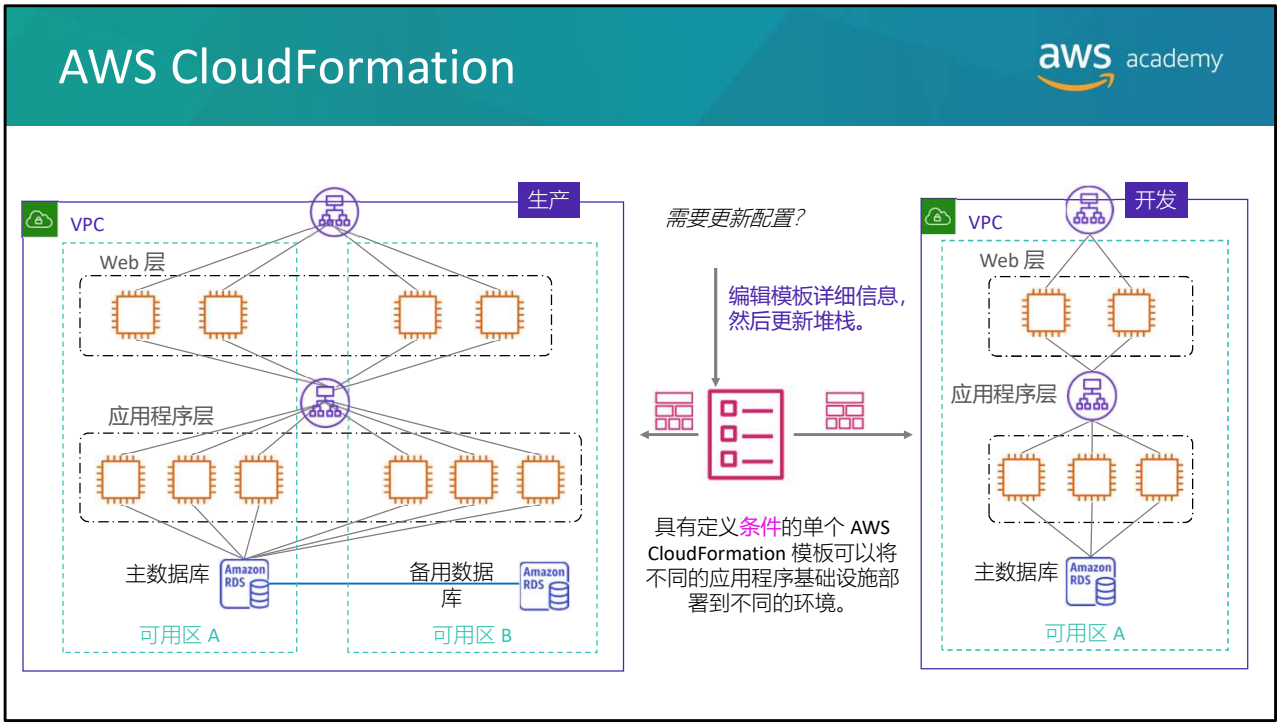



此示例 AWS CloudFormation 模板创建了一个 EC2 实例。尽管此示例并未说明模板的所有可能部分，但它确实突出了一些最常用的部分，包括参数、资源和输出。

**Parameters**（参数）是模板的一个可选部分。参数是在运行时（创建或更新堆栈时）传递给模板的值。您可以从模板的 **Resources**（资源）和**Outputs**（输出）部分引用参数。当用户在控制台中启动 *Create Stack*（创建堆栈）向导时，参数的名称和描述会出现在 *Specify Parameters*（指定参数）页面中。

**Resources**（资源）是任何模板的必填部分。使用它来指定要创建的 AWS 资源及其属性。在此示例中，指定了 *AWS::EC2::Instance* 类型的资源，用它来创建 EC2 实例。示例资源包括静态定义的属性（ImageId *和* InstanceType*）和引用的* KeyPair *参数。*

最后，该示例显示了 **Outputs**（输出）部分。**Outputs**（输出）描述了您查看堆栈的属性时返回的值。在示例中，已声明输出为 InstanceId。创建堆栈后，可在 AWS CloudFormation 控制台的堆栈详细信息中看到此值，方法是运行 *aws cloudformation describe-stacks* AWS 命令行界面 (AWS CLI) 命令，或使用 AWS 软件开发工具包 (SDK) 检索此值。




您可以使用同一 AWS CloudFormation 模板创建生产环境和开发环境。这种方法可确保（例如）在开发和生产中使用相同的应用程序二进制文件、相同的 Java 版本和相同的数据库版本。因此，模板可确保应用程序在生产环境中的行为方式与其在开发环境中的行为方式相同。

在示例中，您可以看到生产环境和开发环境是从同一模板创建的。但生产环境配置为跨两个可用区运行，而开发环境在单个可用区中运行。这些特定于部署的差异可使用*条件*来实现。您可以在 AWS CloudFormation 模板中使用*条件*语句来确保开发、测试和生产环境的配置相同，即使它们的规模和范围有所不同。

您可能需要多个测试环境，用于进行功能测试、用户验收测试和负载测试。手动创建这些环境存在风险。但使用 AWS CloudFormation 创建它们有助于确保一致性和可重复性。


更新堆栈（从而更新 AWS 资源）的一种方法是更新您用于创建堆栈的 AWS CloudFormation 模板，然后运行**更新堆栈**选项。

但是，在实际运行更新之前，您可能希望进一步了解 AWS CloudFormation 在您运行该命令时将实施的特定更改。如果想要进一步了解，可使用*更改集*。

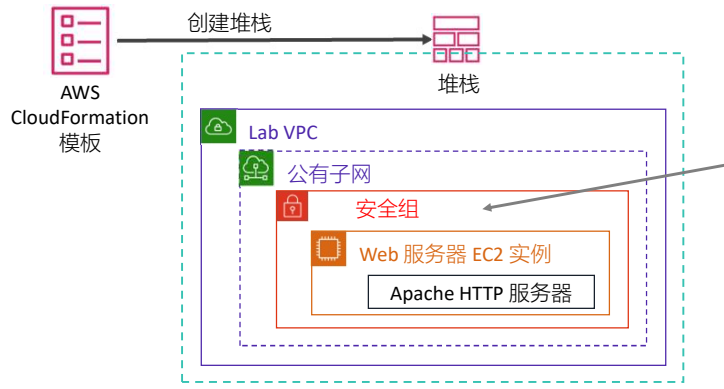
更改集使您可以预览更改，验证它们是否符合您的期望，批准更新之后再继续。

遵循以下基本工作流程使用 AWS CloudFormation 更改集：

1. **创建更改集**：通过为您要更新的堆栈提交更改来实现。
2. **查看更改集**，了解要更改哪些堆栈设置和资源。如果您希望在决定所要进行的更改之前考虑其他更改，请创建其他更改集。
3. **运行更改集**。AWS CloudFormation 会使用这些更改来更新堆栈。

如果您使用更改集，则可能需要对某些资源设置删除策略。DeletionPolicy *属性可用于在删除或更新资源堆栈时保留（或在某些情况下备份）资源。*如果某个资源不具有 DeletionPolicy 属性，AWS CloudFormation 将删除该资源。

# 偏差检测




场景：

1. 应用程序环境由 AWS CloudFormation 堆栈创建。
2. 之后，有人***手动修改安全组*** **并打开新的入站 TCP 端口。**
3. 偏差检测在堆栈上运行。
4. 除安全组之外，所有资源都显示结果 IN_SYNC，安全组显示状态 MODIFIED，并附有详细信息。

问题：在此场景中，如果团队想要修改安全组设置，有什么更好的方法？

答案：修改 AWS CloudFormation 模板安全组设置。然后，**运行更新堆栈**。AWS CloudFormation 将更新安全组。使*模型*部署与实际部署保持同步。

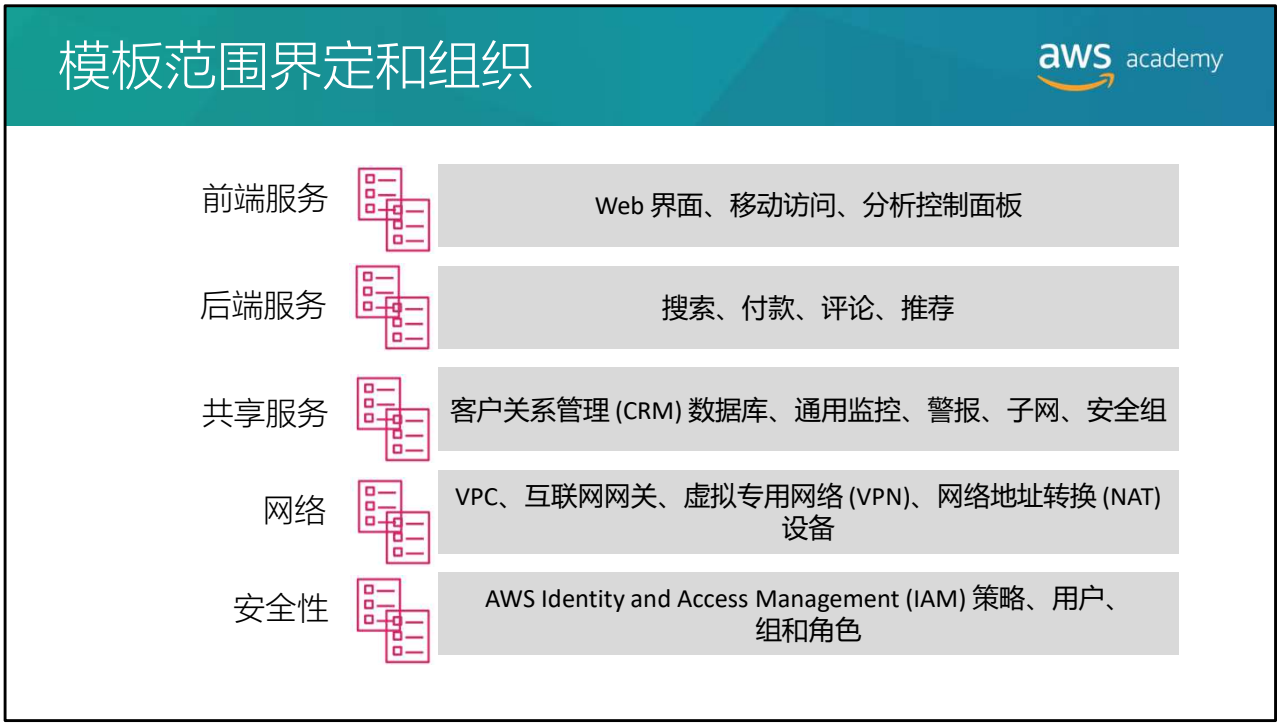
请考虑以下情况：通过运行 AWS CloudFormation 堆栈来创建应用程序环境。然后，某人决定手动修改部署的环境设置。他们在堆栈创建的安全组中创建新的入站规则。但他们是在 AWS CloudFormation 环境外部进行的更改，例如，使用 Amazon EC2 控制台。作为此应用程序的架构师，您想知道您*部署的*环境是否不再匹配 AWS CloudFormation 模板中定义的*模型*环境。

如何知道哪些资源已经过修改，不再完全符合堆栈的规范？

可从控制台中的 **Stack actions**（堆栈操作）菜单选择**Detect Drift**（检测偏差）以对堆栈运行偏差检测。对堆栈执行偏差检测来确定堆栈是否*偏离*预期的模板配置。偏差检测将返回有关在堆栈中支持偏差检测的每个资源的偏差状态的详细信息。

当您删除存在偏差的堆栈时，AWS CloudFormation 的资源清理过程不会处理该偏差。如果堆栈具有未解决的资源依赖项，则可能会导致堆栈删除操作失败。在这种情况下，可能需要手动解决问题。有关详细信息，请参阅支持偏差检测的资源列表。

## 模板范围界定和组织




随着您的组织使用的 AWS CloudFormation 模板越来越多，制定模板策略对您来说非常重要。该策略将定义单个模板应创建的范围，以及使您想要在多个模板中定义 AWS 基础设施的一般特征。

该图提供了一些关于如何整理模板以便维护，并以合理的方式相互组合使用的想法。一个好的策略是将资源定义分组到模板中，类似于将大型企业应用程序的功能整理到不同部分的方式。

考虑基础设施中连接更为紧密的组件，将它们放在相同的模板中。在此示例中，AWS CloudFormation 模板用于在以下五个领域之一创建和维护 AWS 资源：前端服务、后端服务、共享服务、网络和安全性。在每个领域，您都可以维护一个模板，其作用于单个应用程序或单个部门的需求。

无论您如何整理和限定每个 AWS CloudFormation 模板，都要将模板视为需要版本控制的代码。将模板存储在源代码控制系统中。

## AWS Quick Start

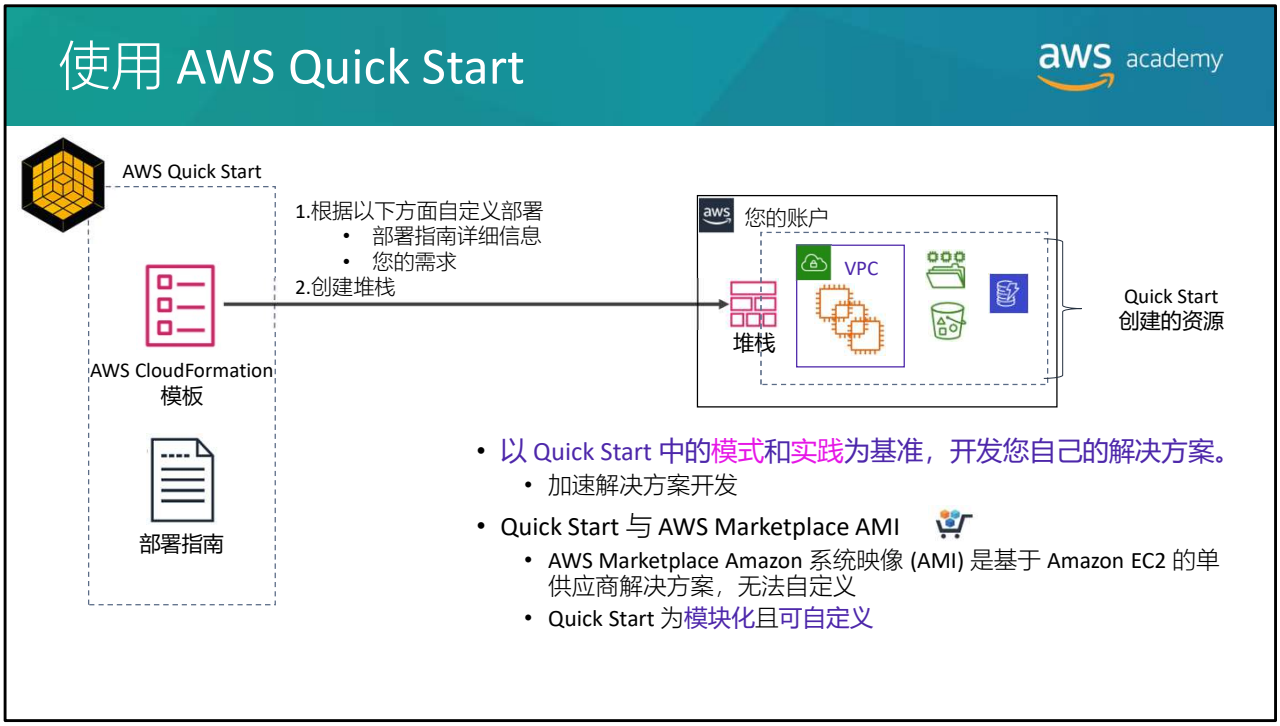
AWS Quick Start

由 AWS 解决方案架构师构建的 AWS CloudFormation 模板

- 黄金标准部署方案
- 遵循安全性和高可用性方面的 AWS 最佳实践
- 一键即可在一小时内创建整个架构
- 可用于实验，也可以作为您自己的架构的基础

AWS Quick Start 提供了 AWS CloudFormation 模板。Quick Start 由 AWS 解决方案架构师和合作伙伴构建，旨在根据 AWS 的安全性和高可用性最佳实践，在 AWS 上部署热门解决方案。这些参考部署可在不到一个小时内在您的 AWS 账户中预置，通常只需几分钟即可完成。它们有助于您通过几个步骤构建架构完善的测试或生产环境。您可以用它们创建整个架构，也可以用它们来试验新的部署方法。

有关详细信息，请转至 AWS Quick Start 页面。




每个 Quick Start 均包含一个 AWS CloudFormation 模板和一个部署指南。该指南提供了有关部署选项以及如何配置部署以满足您的需求的详细信息。

自定义部署以满足您的需求并创建堆栈。根据必须创建的 AWS 资源，Quick Start 将在几分钟或几小时内完成部署。

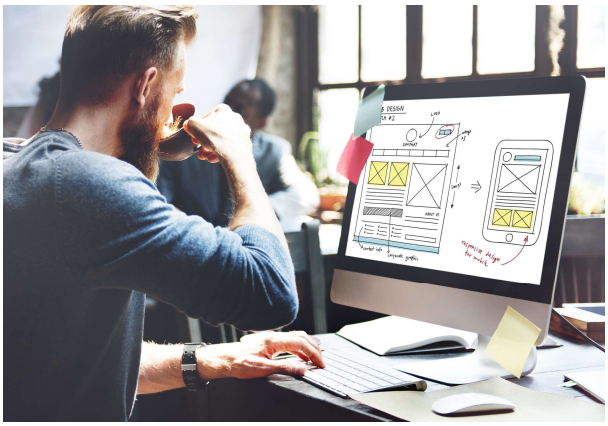
即使不使用 Quick Start，了解 Quick Start 遵循的*模式*和*实践*类型也很有帮助。如果您借鉴 Quick Start 的某个部分并将其嵌入自己的模板，可能有助于加速模板开发。

*AWS Marketplace Amazon 系统映像 (AMI)* 是 人们偶尔会使用的另一种解决方案。可从 Amazon EC2 控制台启动这些资源。AWS Marketplace AMI 提供在 EC2 实例上运行的单一供应商解决方案。相比之下，AWS Quick Start 是模块化的、更可定制的解决方案，可能使用（也可能不使用）Amazon EC2。

# 演示：分析 AWS CloudFormation 模板结构并创建堆栈

讲师可能选择演示 AWS CloudFormation 模板的结构，然后使用模板创建 AWS CloudFormation 堆栈。

# 模块 10 – 指导实验：使用 AWS CloudFormation 实现自动化基础设施部署



您现在将完成“模块 10 – 指导实验：使用 AWS CloudFormation 实现自动化基础设施部署”。

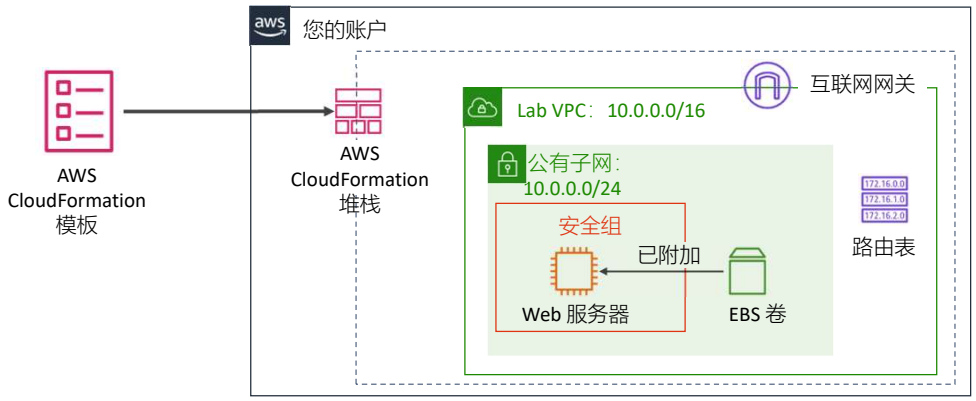
# 指导实验：任务

1. 部署网络层
2. 部署应用程序层
3. 更新堆栈
4. 使用 AWS CloudFormation Designer 浏览模板
5. 删除堆栈

在本指导实验中，您将完成以下任务：

1. 部署网络层
2. 部署应用程序层
3. 更新堆栈
4. 使用 AWS CloudFormation Designer 浏览模板
5. 删除堆栈

# 指导实验：最终产品




在本指导实验结束时，您将使用 AWS CloudFormation 在图中创建资源。网络层中的资源是您在创建第一个堆栈时创建的。EC2 实例、安全组和 Amazon Elastic Block Store (Amazon EBS) 卷是您在创建第二个堆栈时创建的。然后，在用于创建第二个堆栈的模板中更新安全组设置。当您运行更新堆栈操作时，此修改将应用于安全组资源。


现在可以开始指导实验了。

# 指导实验总结：要点

完成这个指导实验之后，您的讲师可能会带您讨论此指导实验的要点。

# 第 3 节要点


aws academy

- AWS CloudFormation 是一项基础设施即代码 (IaC) 服务，使用此服务，您可以建模、创建和管理 AWS 资源的集合
- AWS CloudFormation IaC 在使用 JSON 或 YAML 编写的模板中定义
- 堆栈是您在使用模板创建 AWS 资源时创建的
- 可对现有堆栈执行的操作包括更新堆栈、检测偏差和删除堆栈
- AWS Quick Start 提供了解决方案架构师构建的反映 AWS 最佳实践的 AWS CloudFormation 模板

本模块中这节内容的要点包括：

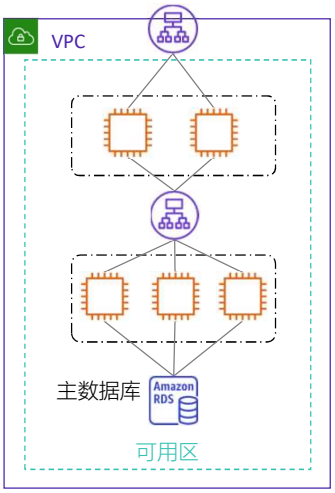
- AWS CloudFormation 是一项基础设施即代码 (IaC) 服务，使用此服务，您可以建模、创建和管理 AWS 资源的集合
- AWS CloudFormation IaC 在使用 JSON 或 YAML 编写的模板中定义
- 堆栈是您在使用模板创建 AWS 资源时创建的
- 可对现有堆栈执行的操作包括更新堆栈、检测偏差和删除堆栈
- AWS Quick Start 提供了解决方案架构师构建的反映 AWS 最佳实践的 AWS CloudFormation 模板

模块 10：实现架构自动化

# 第 4 节：实现部署自动化

介绍第 4 节：实现部署自动化。

## 如何保持队列更新?


您可能要管理数百个实例。
如何应用访客操作系统 (OS) 补丁并更新安装在其上的软件?

在上一节中，您学习了如何使用 AWS CloudFormation 自动创建整个基础设施。其功能非常强大，但仍有些重要问题需要解决。

如何更新 EC2 实例队列中的软件？您是否要自行远程登录每个实例并运行更新命令？如果出现错误，如何还原更改？如果您有成百上千台运行许多不同应用程序的服务器怎么办？

传统工具可以帮助应对这些情况，但现成的解决方案会更加方便。

## AWS Systems Manager

*了解运行状况并对 AWS 资源执行操作。*


- 自动执行操作任务
  - 示例：在 EC2 实例队列中应用操作系统补丁和软件升级
- 简化资源和应用程序管理
  - 管理软件清单
  - 查看整个队列的详细系统配置
- 在本地和云中管理服务器

即使您使用像 AWS CloudFormation 这样的 IaC 工具来创建和维护 AWS 资源部署，拥有其他可用的工具也很有帮助。例如，这些工具可以满足环境对配置管理的持续需求。这些需求可能发生在预置基础设施资源之后和基础设施启动并运行之后。AWS Systems Manager 服务可解决这一难题。

AWS Systems Manager 是一项管理服务，高度专注于自动化。它允许配置和管理在本地或 AWS 中运行的系统。AWS Systems Manager 使您能够识别要管理的实例，然后定义要对这些实例执行的管理任务。您可以免费使用 AWS Systems Manager 来管理您的 Amazon EC2 和本地资源。

您可以使用 AWS Systems Manager 完成的一些任务包括：

- 收集软件清单
- 应用操作系统 (OS) 补丁
- 创建系统映像
- 配置 Microsoft Windows 和 Linux 操作系统

这些功能可帮助您定义和跟踪系统配置，防止偏差，并确保 Amazon EC2 和本地配置的软件合规性。


本示例说明了如何使用 Systems Manager 来更新、管理和配置 EC2 实例队列。

您可以在 EC2 实例上，甚至在本地服务器或虚拟机 (VM) 上安装 *AWS Systems Manager 代理（SSM 代理）*。在安装 SSM 代理后，Systems Manager 就可以更新、管理和配置它所安装到的服务器。代理处理来自 Systems Manager 的请求，然后根据请求中提供的规范运行它们。然后，代理将状态和相关信息发回 Systems Manager。

默认情况下，大多数 Microsoft Windows Server AMI、所有 Amazon Linux 和 Amazon Linux 2 AMI 以及某些 Ubuntu AMI 上都会预安装 SSM 代理。但您必须在从其他 Linux AMI 创建的 EC2 实例上手动安装代理。有关完整详细信息，请参阅使用 SSM 代理 AWS 文档。

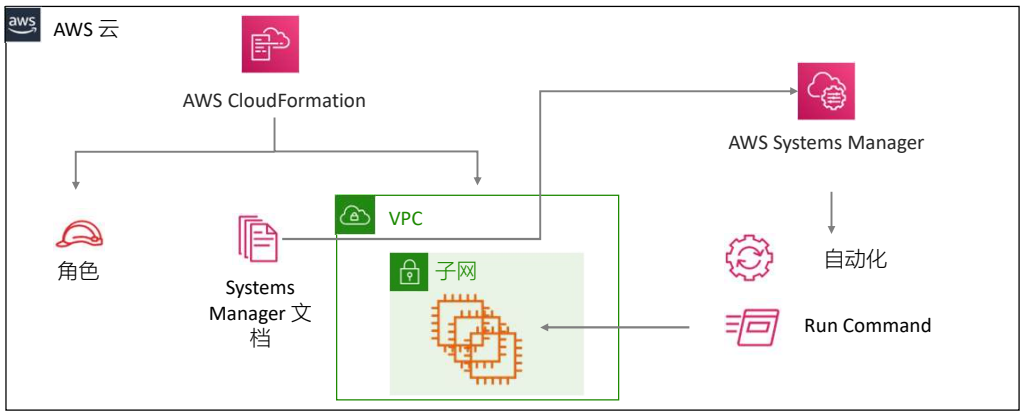
AWS Systems Manager 提供了各种工具：

- **Run Command** 使您能够以远程方式安全地管理托管实例的配置。命令可以在没有 Secure Shell (SSH) 或远程桌面协议 (RDP) 访问的情况下运行，因此您可以使用它们来减少对堡垒主机的需求。运行 Bash、PowerShell、Salt 或 Ansible 脚本。
- **维护时段**使您能够制定计划，对您的实例执行潜在的破坏性操作。示例包括修补操作系统、更新驱动程序或安装软件或补丁。
- **参数仓库**可提供安全存储，用于配置数据和密钥管理。例如，您可以将密码、数据库字符串和许可证代码存储为参数值。
- **补丁管理器**可自动执行通过安全相关的更新以及其他类型的更新修补托管实例的流程。
- **状态管理器**可自动执行将您的 Amazon EC2 和混合基础设施保持在您定义的状态的过程。
- **Automation** 使您能够构建自动化工作流程以配置和管理实例和 AWS 资源。

- **Session Manager** 使您能够通过基于交互式浏览器的 shell 管理 EC2 实例。
- **Inventory** 使您可以了解 Amazon EC2 和本地计算环境。您可以使用 Inventory 功能从托管实例收集元数据。
- **文档**定义了 Systems Manager 对您的托管实例执行的操作。您可以通过在运行时指定参数来使用十多个预先配置的文档。您还可以定义 JSON 或 YAML 格式的文档，并指定步骤和参数。

## AWS CloudFormation 和 Systems Manager 相互补充

AWS CloudFormation 适用于定义 AWS 云资源。
Systems Manager 适用于在访客操作系统内实现自动化。




现在，您已了解 AWS CloudFormation 和 AWS Systems Manager 的功能，请考虑这两种服务如何互补。

Systems Manager 适用于在访客操作系统*内*实现自动化。相比之下，AWS CloudFormation 适用于定义 AWS 云资源。

您可以在 AWS 云层使用 AWS CloudFormation 来定义 AWS 资源。然后，如图所示，您可以使用 AWS Systems Manager 配置由 AWS CloudFormation 堆栈创建的实例的操作系统。

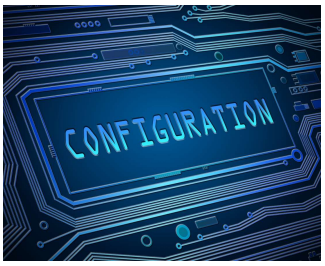
通过使用 AWS CloudFormation 维护云资源，您可以保持部署堆栈，然后从单个模板中删除堆栈的能力。Systems Manager 则为您提供了一种执行持续任务的方法，例如，通过补丁更新来更新 EC2 实例访客操作系统，将日志集中聚合到 Amazon CloudWatch。

有关将这两项服务结合使用的示例解决方案的更多信息，请参阅 Using AWS Systems Manager Automation and AWS CloudFormation together 博客文章。

## AWS OpsWorks


AWS OpsWorks 是一项配置管理服务。

- 实现服务器配置、部署和管理方式的自动化
- 提供 Chef 和 Puppet 的托管实例
  - Chef 和 Puppet 是常用的自动化平台
- 三个版本可供选择 –
  - AWS OpsWorks for Chef Automate
  - AWS OpsWorks for Puppet Enterprise
  - AWS OpsWorks Stacks



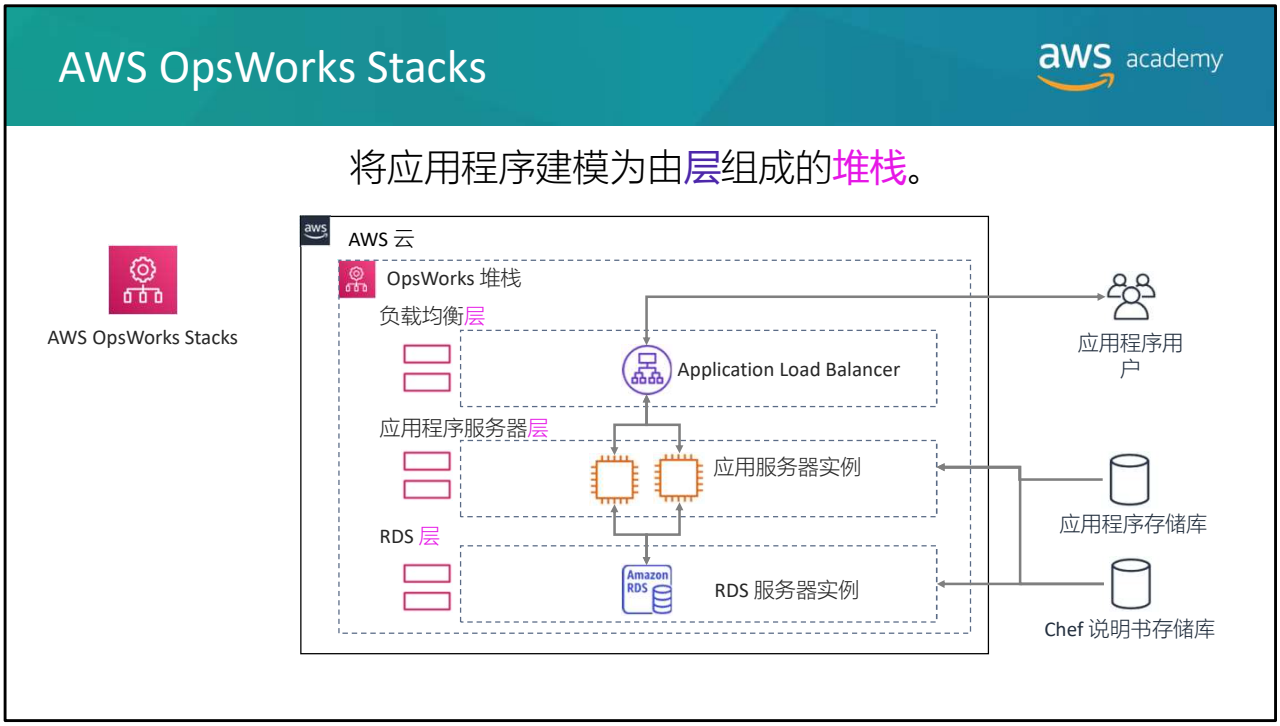

*AWS OpsWorks* 是一项配置管理服务。您可以使用 OpsWorks 来自动配置、部署和管理 EC2 实例。

AWS OpsWorks 有三种不同的版本：

- **AWS OpsWorks for Chef Automate** 提供了一个完全托管的 Chef Automate 服务器，该服务器针对持续部署提供了工作流程自动化，并针对合规性和安全性提供了自动化测试。Chef Automate 平台可以处理软件与操作系统配置、持续合规性、软件包安装和数据库设置等多种操作任务。您可以使用 Chef Automate 来创建和管理基于 AWS 云运行的动态基础设施。Chef Automate 服务器通过告知 Chef 客户端在节点上运行哪些 Chef 配方来管理环境中节点的配置。它还会存储有关节点的信息，并作为 Chef 说明书的中央存储库。

- **AWS OpsWorks for Puppet Enterprise** 提供了一个托管的 Puppet Enterprise 服务器和一套自动化工具，允许针对编排、自动预置和可追踪性可视化实现工作流程自动化。借助 Puppet Enterprise，您能够以支持维护和版本控制（类似应用程序源代码）的方式定义服务器的配置。主 Puppet 服务器旨在一致地配置和维护您的其他 Puppet 服务器（或节点）。您也可以根据其他节点的状态动态配置您的节点。

- **AWS OpsWorks Stacks** 是一项配置管理服务，可帮助您利用 Chef 配置和操作各种类型和规模的应用程序。您可以定义应用程序的架构和每个组件的规格，包括软件包安装、软件配置和存储等资源。




该示例演示了如何使用 AWS OpsWorks Stacks 管理基本应用程序。OpsWorks Stacks 应用程序中的基本创建单位是*堆栈*。

创建堆栈后，您可以向该堆栈添加多个*层*。因此，您可以将应用程序构建为一组相关功能的交互层。

在这种情况下，一组应用程序服务器在*应用程序服务器层*中运行。这些应用程序在*负载均衡层*中定义的 Elastic Load Balancing 负载均衡器之后运行。该示例还包括在 *RDS 层*中定义的后端 Amazon Relational Database Service (Amazon RDS) 数据库服务器。

层依靠 Chef 配方来处理诸如在实例上安装软件包、部署应用程序、运行脚本等任务。OpsWorks Stacks 使用 Chef 说明书处理安装和配置软件包和部署应用程序等任务。 您的自定义说明书必须存储在在线存储库中，它或者是一个存档（如 .zip 文件），或者是一个源代码控制管理器（如 Git）。

OpsWorks Stacks 的一项重要功能是一组*生命周期事件*（包括设置、配置、部署、取消部署和关闭），这些事件会在适当的时间自动在每个实例上运行一组指定的配方。

每个层都可以有一组分配给每个生命周期事件的配方。这些配方可处理该事件和层的各种任务。

## OpsWorks Stacks 是 AWS CloudFormation 的补充

1.使用 AWS CloudFormation 创建基础设施（VPC、IAM 角色等）。

2.使用 OpsWorks Stacks 部署应用程序层。


由于可通过 AWS CloudFormation 创建 OpsWorks Stacks，因此这两种技术可以互补使用。

例如，您可以使用一个 AWS CloudFormation 模板为您的环境创建 AWS 资源基础设施，包括 VPC。然后，您可以使用另一个 AWS CloudFormation 模板创建将在该 VPC 中部署的 OpsWorks 堆栈。在您的账户中创建两个 AWS CloudFormation 堆栈之后，您可以使用 OpsWorks 堆栈来管理应用程序。

模块 10：实现架构自动化

# 第 5 节：AWS Elastic Beanstalk

介绍第 5 节：AWS Elastic Beanstalk。

# 一般性挑战

围绕应用程序部署管理基础基础设施可能*非常困难*

管理和配置服务器可能*非常耗时*

多个项目或应用程序之间可能*缺乏一致性*

现在，考虑在着手管理云基础设施时可能会遇到的一些一般性挑战。

首先，部署应用程序可能*非常困难*。如何确保它高度可用，即使在高峰使用期间也能支持用户请求？如何确保应用程序具有弹性，并出于灾难恢复 (DR) 目的定期备份？管理和配置服务器可能*非常耗时*。同时，您希望在项目和应用程序之间保持*一致性*，但这种一致性可能很难实现。

# AWS Elastic Beanstalk


- 启动并运行 Web 应用程序的简单方式
- 可自动处理以下任务的托管服务 –
  - 基础设施预置和配置
  - 部署
  - 负载均衡
  - 自动扩展
  - 运行状况监控
  - 分析和调试
  - 日志记录
- 无需额外付费即可使用
  - 只需为您使用的底层资源付费

AWS Elastic Beanstalk 是另一种 AWS 计算服务选项。它是一种平台即服务 (PaaS) 产品，有助于快速部署、扩展和管理您的 Web 应用程序和服务。它能够解决前面提到的许多难题。

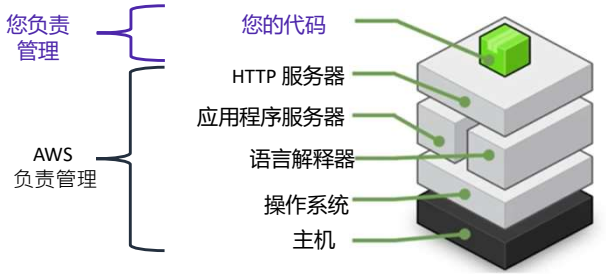
借助 Elastic Beanstalk，您可以保持对代码的控制，由 AWS 维护底层基础设施。使用 AWS 管理控制台中的简单向导创建和部署所需的 AWS 资源。该向导要求您选择实例类型和大小、数据库类型和大小以及要使用的自动扩展设置。它允许您访问服务器日志文件，并在负载均衡器上启用安全 HTTP (HTTPS)。

在您上传代码后，Elastic Beanstalk 将自动处理包括容量预置、负载均衡、自动扩展和应用程序运行状况监控在内的部署工作。同时，您保留了对支持您的应用程序的 AWS 资源的完全控制，并且可以随时访问底层资源。

AWS Elastic Beanstalk 不收取额外费用。您只需为您创建的用于存储和运行应用程序的 AWS 资源付费，例如 EC2 实例或 S3 存储桶。您只需在使用时为您的实际用量支付费用。

## AWS Elastic Beanstalk 部署

- 它支持为通用平台编写的 Web 应用程序
  - Java、.NET、PHP、Node.js、Python、Ruby、Go 和 Docker
- 上传您的代码
  - Elastic Beanstalk 将自动处理部署
  - 部署在 Apache、NGINX、Passenger、Puma 和 Microsoft Internet 信息服务 (IIS) 等服务器上

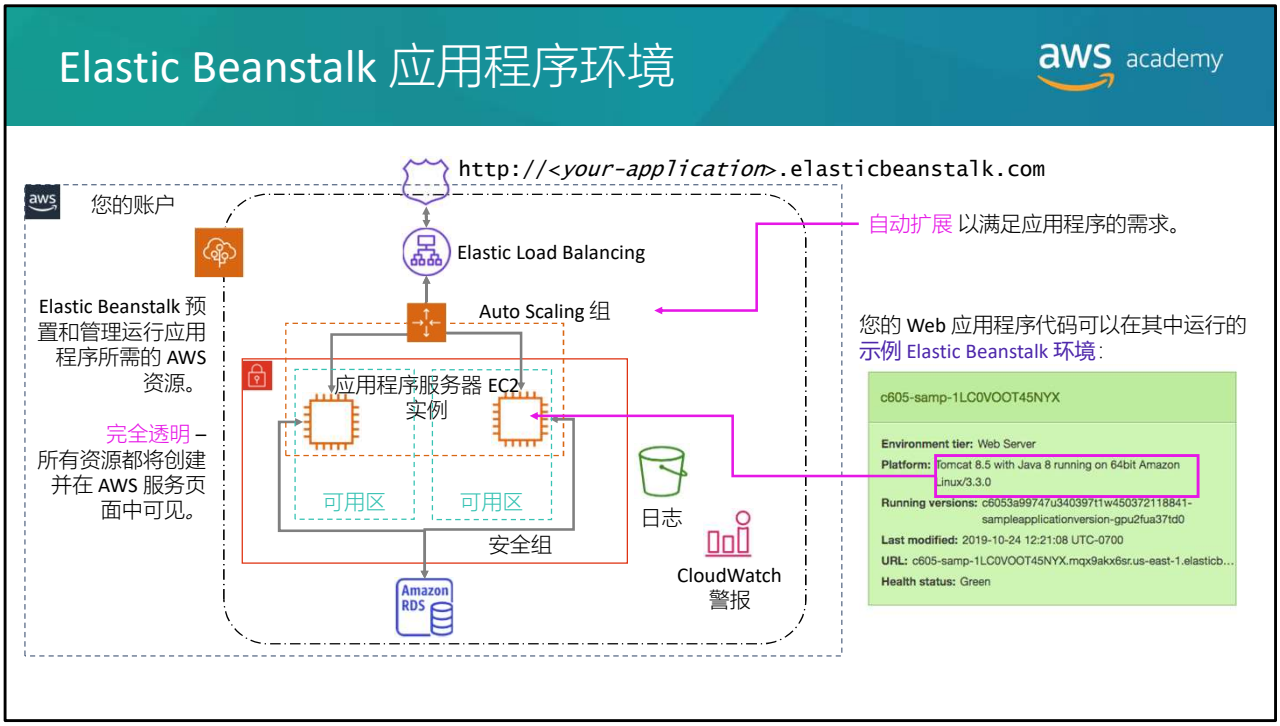



Elastic Beanstalk 会使用在选定平台上运行应用程序所需的必要组件来配置您环境中的每个 EC2 实例。您无需负责登录实例来安装和配置应用程序堆栈。

您只需创建代码。Elastic Beanstalk 旨在使您的应用程序部署变得快速而简单。它支持一系列平台，包括 **Docker**、**Go**、**Java**、**.NET**、**Node.js**、**PHP**、**Python** 和 **Ruby**。

AWS Elastic Beanstalk 将您的代码部署在：

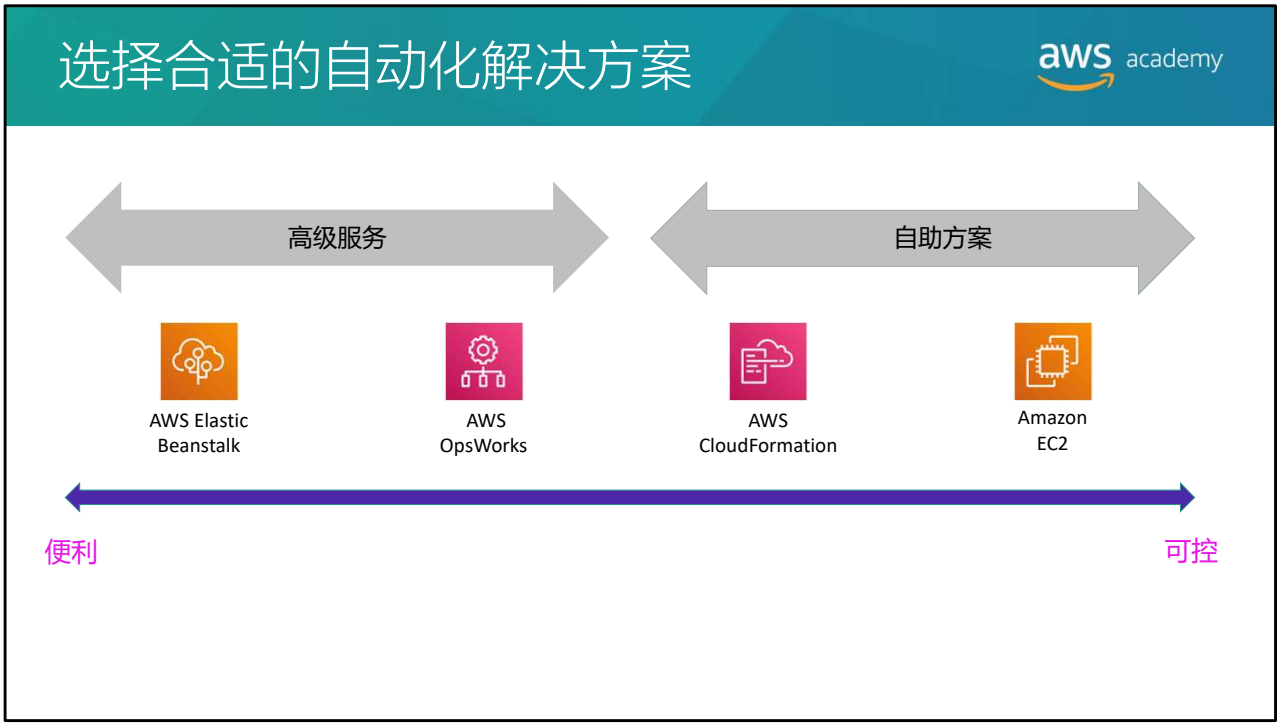
- **Apache Tomcat**，适用于 Java 应用程序
- **Apache HTTP Server**，适用于 PHP 和 Python 应用程序
- **NGINX** 或 **Apache HTTP Server**，适用于 Node.js 应用程序
- **Passenger** 或 **Puma**，适用于 Ruby 应用程序
- **Microsoft Internet Information Services (IIS)**，适用于 .NET、Java SE、Docker 和 Go 应用程序。




使用 Elastic Beanstalk 时，您可以从两种类型的环境中选择。单实例环境*允许您启动单个 EC2 实例，且不包含负载均衡或自动扩缩。*另一种类型的环境（本例所示）可以启动*多个 EC2 实例*，且包含负载均衡和自动扩缩配置。托管数据库层是可选的。

由 Elastic Beanstalk 创建的 AWS 资源会显示在您的 AWS 账户中。例如，创建 Elastic Beanstalk 应用程序后，打开 AWS 管理控制台，然后打开 Amazon EC2 控制台。您将看到 Elastic Beanstalk 代表您管理的实例。在本例中，我们创建了两个 EC2 实例。每个 EC2 实例都运行 Amazon Linux 访客操作系统，并安装了 Java 8 和 Apache Tomcat 8.5 Web 服务器。您的应用程序代码将在这些服务器上运行。配置了自动扩缩功能，因此，如果负载开始使这两个实例上的资源紧张（例如，过高的 CPU 利用率持续 5 分钟以上），则系统会自动启动更多的应用程序服务器实例。本示例还显示了 RDS 数据库实例可用，并可从 EC2 实例访问。您可以将应用程序数据存储在数据库中，并在应用程序代码中使用结构化查询语言 (SQL) 来访问和更新这些数据。Elastic Beanstalk 管理数据库实例，并帮助维护 EC2 实例和数据库之间的连接。

Elastic Beanstalk 创建和管理可扩展的环境。您可以将 Auto Scaling 组配置为自动扩展应用程序，以处理大量流量负载。它还为您的应用程序环境提供唯一的域名。URL 语法是 <*your-application*>.elasticbeanstalk.com。您也可以使用 Amazon Route 53 将自己的域名解析为提供的域名。




本模块至少向您介绍了四种 AWS 服务。一个常见问题是，关于提供应用程序管理功能的多种服务，它们之间的界线是什么，或者哪种服务应在什么情况下使用？您的决定应取决于所需的相对便利和控制水平。

*Elastic Beanstalk* 是一项易于使用的应用程序服务，用于构建在 Java、PHP、Node.js、Python、Ruby 或 Docker 上运行的 Web 应用程序。如果您希望无需自定义环境即可上传代码，那么 Elastic Beanstalk 可能是一个不错的选择。

*OpsWorks* 让您可以启动应用程序并定义其架构和每个组件的规格，包括软件包安装、软件配置和资源（比如存储）。您可以使用用于常见技术的模板（应用程序服务器、数据库等），也可以构建自己的模板。

相较编写和维护 AWS CloudFormation 模板以创建堆栈或直接管理 EC2 实例，Elastic Beanstalk 和 OpsWorks 提供了更高级别的服务。但正确选择要使用的服务（或服务组合）取决于您的需求。这些工具都可供您使用。作为架构师，您必须确定哪些服务最适合您的使用案例。

# 第 5 节要点

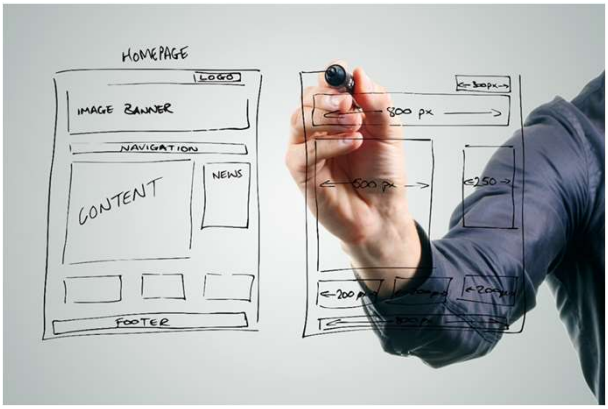

aws academy

- Elastic Beanstalk 可创建和管理可扩缩且高度可用的 Web 应用程序环境，使您能够专注于应用程序代码
- 您可以使用 Java、.NET、PHP、Node.js、Python、Ruby、Go 或 Docker 编写 Elastic Beanstalk 应用程序代码
- 由 Elastic Beanstalk 创建的 AWS 资源是完全透明的 – 在 AWS 管理控制台服务页面视图中可见
- Elastic Beanstalk 不收取额外费用 – 您只需为使用的底层资源付费

本模块中这节内容的要点包括：

- AWS Elastic Beanstalk 用于创建和管理可扩展且高度可用的 Web 应用程序环境，使您能够专注于应用程序代码
- 您可以使用 Java、.NET、PHP、Node.js、Python、Ruby、Go 或 Docker 编写 Elastic Beanstalk 应用程序代码
- 由 Elastic Beanstalk 创建的 AWS 资源是完全透明的 – 在 AWS 管理控制台服务页面视图中可见
- Elastic Beanstalk 不收取额外费用 – 您只需为使用的底层资源付费

# 模块 10 – 挑战实验：实现基础设施部署自动化




您现在将完成“模块 10 – 挑战实验：实现基础设施部署自动化”。

# 业务需求：实施 IaC

- 咖啡馆现在在多个国家/地区设有分店，必须开始自动化才能保持发展。
- 他们需要一种方法，能够跨多个 AWS 服务一致地部署、管理和更新咖啡馆资源。
- 他们希望能够跨 AWS 区域可靠地创建可重复的环境，以满足开发和生产需求。

长期以来，咖啡馆一直在创建 AWS 资源并手动配置应用程序。这种方法可以有效帮助咖啡馆快速开发 Web 服务，构建支持员工和客户需求的基础设施。但是，他们发现将部署复制到新的 AWS 区域，以支持在多个国家/地区设立的咖啡馆分店挑战重重。

咖啡馆还希望拥有具有可靠匹配配置的独立开发和生产环境。他们意识到，必须开始自动化才能支持持续增发展。

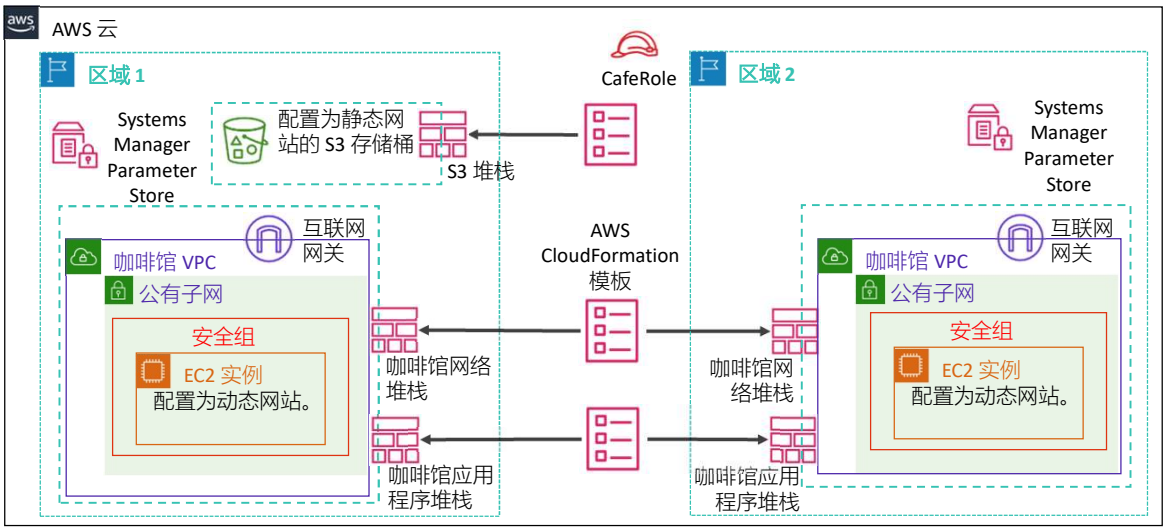
# 挑战实验：任务

1. 从头创建 AWS CloudFormation 模板
2. 将存储桶配置为网站并更新堆栈
3. 克隆包含 AWS CloudFormation 模板的 CodeCommit 存储库
4. 使用 AWS CloudFormation、CodeCommit 和 CodePipeline 创建新的网络层
5. 更新网络堆栈
6. 定义 EC2 实例资源并创建应用程序堆栈
7. 将咖啡馆网络和网站复制到另一个 AWS 区域

在本挑战实验中，您将完成以下任务：

1. 从头创建 AWS CloudFormation 模板
2. 将存储桶配置为网站并更新堆栈
3. 克隆包含 AWS CloudFormation 模板的 CodeCommit 存储库
4. 使用 AWS CloudFormation、CodeCommit 和 CodePipeline 创建新的网络层
5. 更新网络堆栈
6. 定义 EC2 实例资源并创建应用程序堆栈
7. 将咖啡馆网络和网站复制到另一个 AWS 区域

# 挑战实验：最终产品




该图显示了您将在挑战实验中构建的完整架构。


现在可以开始挑战实验了。

# 挑战实验总结：要点

完成这个指导实验之后，您的讲师可能会带您讨论此指导实验的要点。

模块 10：实现架构自动化

# 模块总结

现在来回顾下本模块，并对知识测验和对实践认证考试问题的讨论进行总结。

# 模块总结

aws academy

总体来说，您在本模块中学习了如何：

- 识别何时实现自动化及原因
- 确定如何使用 AWS CloudFormation 建模、创建和管理 AWS 资源集合
- 使用 Quick Start AWS CloudFormation 模板设置架构
- 说明如何借助 AWS System Manager 和 AWS OpsWorks 实现基础设施和部署自动化
- 说明如何使用 AWS Elastic Beanstalk 部署简单的应用程序

总体来说，您在本模块中学习了如何：

- 识别何时实现自动化及原因
- 确定如何使用 AWS CloudFormation 建模、创建和管理 AWS 资源集合
- 使用 Quick Start AWS CloudFormation 模板设置架构
- 说明如何借助 AWS System Manager 和 AWS OpsWorks 实现基础设施和部署自动化
- 说明如何使用 AWS Elastic Beanstalk 部署简单的应用程序

# 完成知识测验


现在可以完成本模块的知识测验。

# 样题

考虑这样一种情况，您希望创建单一的 AWS CloudFormation 模板，该模板既能创建跨两个可用区的生产环境，又能创建存在于单个可用区中的开发环境。

您希望利用 AWS CloudFormation 模板的哪个可选部分来配置支持此功能的逻辑？

A. 资源
B. 输出
C. 条件
D. 描述

请查看答案选项，并根据之前突出显示的关键字排除错误选项。

**正确答案是 C – 条件。**可选的**条件**部分包含一些声明，定义了在哪些情况下创建或配置实体。**资源**不是 AWS CloudFormation 模板的可选部分。输出**部分不会影响堆栈运行时模板将部署的 AWS 资源数量。**描述**部分不影响配置。**

# 其他资源

- AWS 上的部署选项概览
- 使用 AWS CloudFormation 模板
- AWS CloudFormation 示例模板
- AWS OpsWorks Stacks 常见问题
- AWS Systems Manager 功能
- AWS Elastic Beanstalk 常见问题

如果您想了解有关本模块所涵盖主题的更多信息，下面这些其他资源可能会有所帮助：

- AWS 上的部署选项概览
- 使用 AWS CloudFormation 模板
- AWS CloudFormation 示例模板
- AWS OpsWorks Stacks 常见问题
- AWS Systems Manager 功能
- AWS Elastic Beanstalk 常见问题


感谢您完成本模块的学习。