Red Hat Ansible Automation Platform 2.4

在 Red Hat OpenShift 上部署 Ansible Automation Platform 2

Last Updated: 2023-11-16

Red Hat Ansible Automation Platform 2.4 在 Red Hat OpenShift 上部署 Ansible Automation Platform 2

Roger Lopez
ansible-feedback@redhat.com

法律通告

摘要

本文档提供了在 Red Hat OpenShift 上部署 Ansible Automation Platform 2 的最佳实践。

目录

注释和反馈 3

第 1 章 概述 4

第 2 章 为什么选择 RED HAT OPENSHIFT 上的 ANSIBLE AUTOMATION PLATFORM？ 6

第 3 章 开始前 7
3.1. POD 和容器的资源管理 7
3.2. 自动化控制器 POD 容器的大小建议 8
3.3. POSTGRES POD 的大小建议 10
3.4. 自动化作业 POD 的大小建议 10
3.5. 自动化控制器 POD 大小建议概述 12
3.6. AUTOMATION HUB POD 的大小建议 12
3.7. 为自动化控制器 POD 指定专用节点 14
3.8. 处理数据库高可用性 15

第 4 章 先决条件 16

第 5 章 安装 ANSIBLE AUTOMATION PLATFORM OPERATOR 17

第 6 章 安装自动化控制器 18

第 7 章 安装自动化中心 20

第 8 章 登录到您的自动化控制器仪表板 22

第 9 章 登录到您的自动化中心仪表板 23

第 10 章 监控 ANSIBLE AUTOMATION PLATFORM 24
10.1. 哪些将用于监控 API 指标？ 24
10.2. 我应该查看什么指标？ 24
10.3. 通过 ANSIBLE PLAYBOOK 安装 24

附录 A. 关于 AUTHOR 27

附录 B. 从以前的 AAP 安装中删除现有的 PVC 28

附录 C. 将标签和污点应用到 RED HAT OPENSHIFT 节点 29

附录 D. 创建 AMAZON S3 存储桶 30

附录 E. 创建 AWS S3 SECRET 31

附录 F. 在 AAP OPERATOR 中添加额外的受管命名空间 32

附录 G. 参考 33

附录 H. REVISION HISTORY 34

注释和反馈

在开源的感觉中，我们邀请任何人提供有关任何参考架构的反馈和评论。虽然我们在内部回顾了我们的文档，但有时会遇到问题或排字错误。反馈使我们不仅能提高我们生成的文档质量，而是让读者能够向其潜在改进和主题扩展提供自己的想法。您可以通过发送电子邮件至 ansible-feedback@redhat.com 来提供关于文档的反馈。请参阅电子邮件中的标题。

第 1 章 概述

Red Hat OpenShift 参考架构上的 Ansible Automation Platform (AAP) 2.3 提供了部署 Ansible Automation Platform 环境的建议设置。它提供逐步部署过程，附带安装和配置 Ansible Automation Platform 2.3 的最新最佳实践。它最适合希望在 Red Hat OpenShift 上部署 Ansible Automation Platform 的系统和平台管理员。

通过使用 Red Hat OpenShift 的强大功能，我们可以简化 Ansible Automation Platform 部署，并显著减少设置它所需的时间和工作量。

图 1.1. 自动化控制器架构

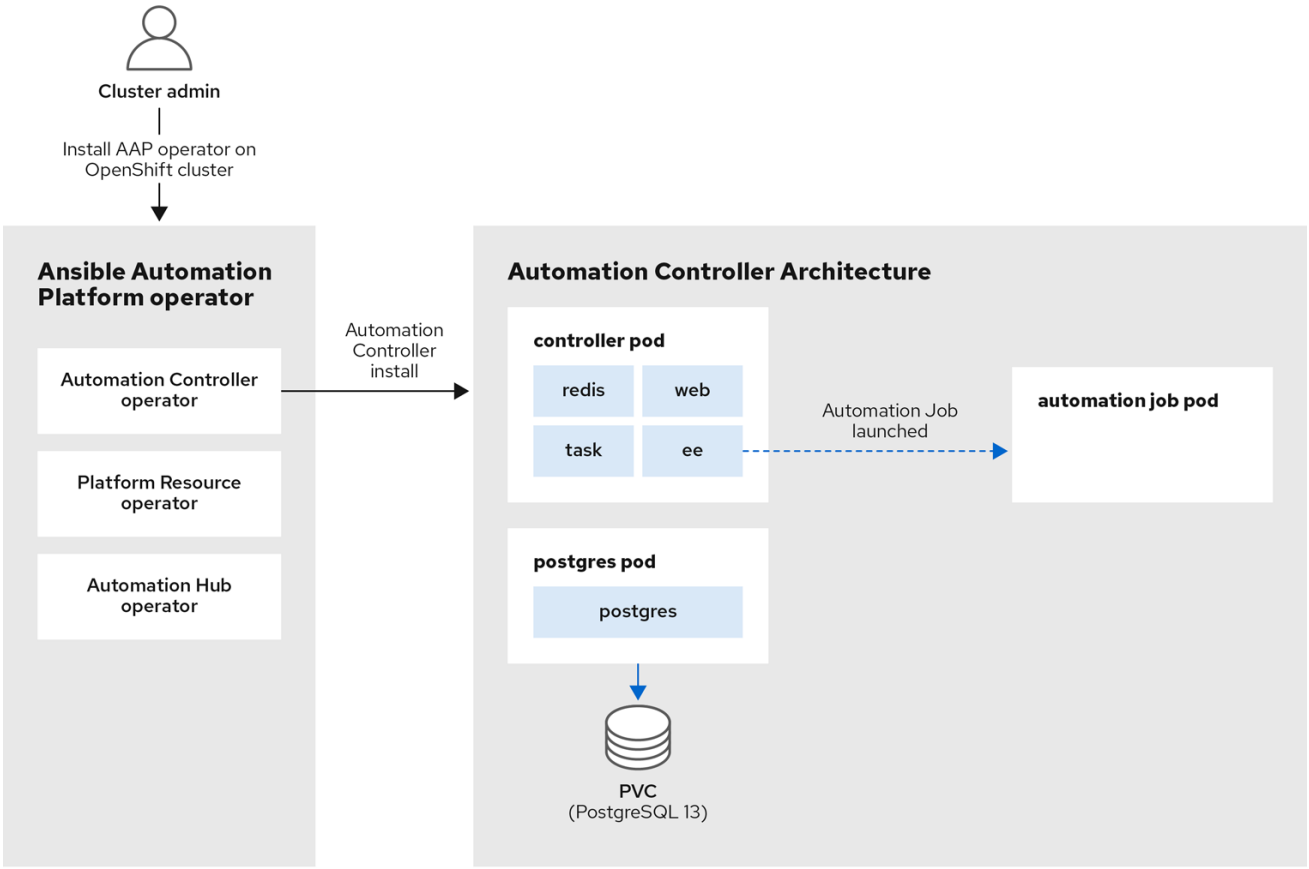
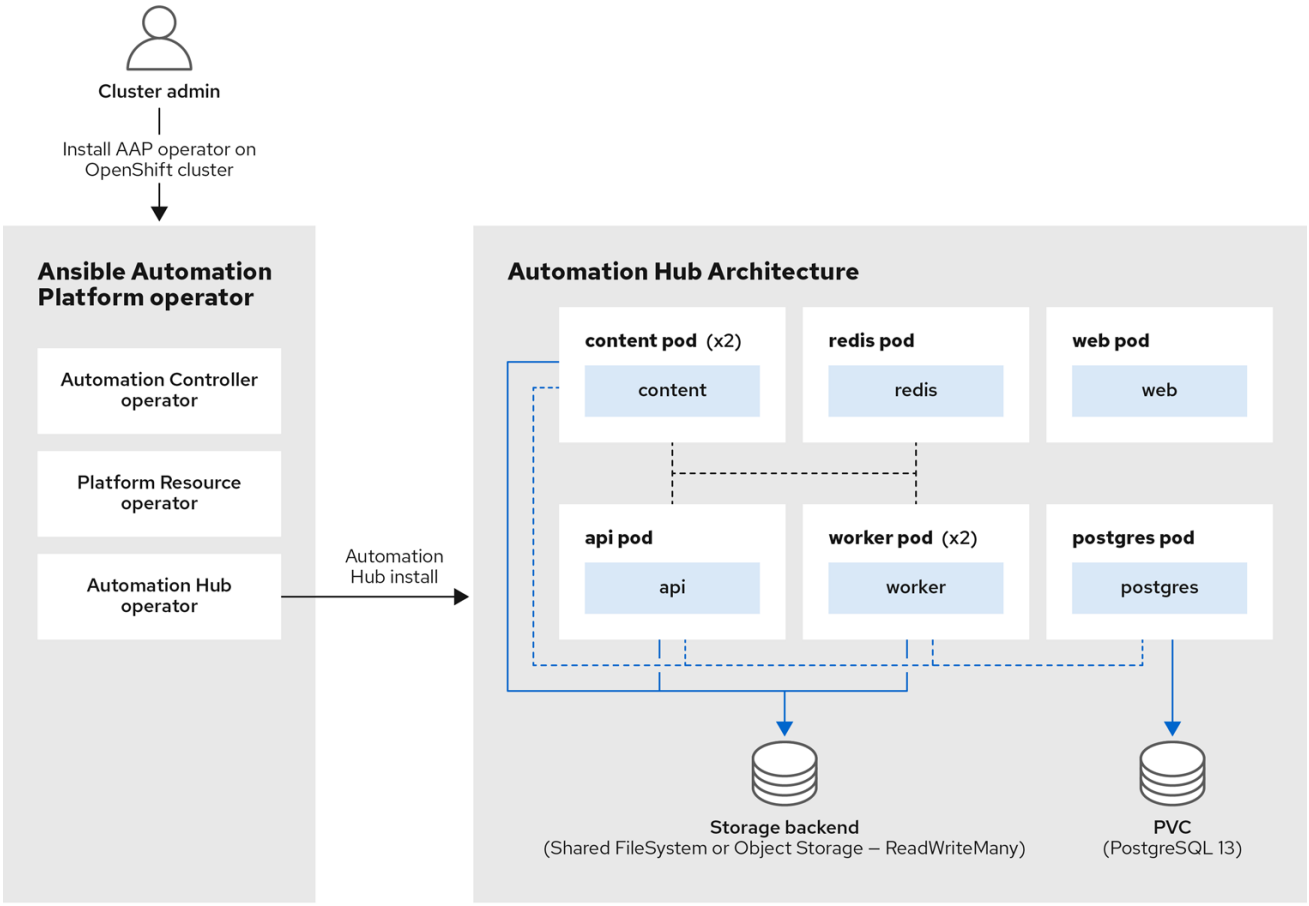


图 1.1 “自动化控制器架构” 显示 Ansible Automation Platform (AAP) Operator 部署自动化控制器组件的部署过程流。自动化控制器 Operator 是组成更大 Ansible Automation Platform operator 的三个操作器之一，它负责部署各种 pod，包括控制器、postgres 和自动化作业 pod。

图 1.2. Automation hub 架构



同样，图 1.2 “Automation hub 架构” 显示 AAP Operator 部署自动化中心组件。Automation hub operator 部署各种 pod，它们相互通信，以提供自动化中心，以共享内部生成的内容、Red Hat Ansible 认证的内容、执行环境和 Ansible 验证的内容与团队一起。

此外，此参考架构突出显示了为任何自动化工作提供高效、可扩展的环境所涉及的关键步骤。

第 2 章 为什么选择 RED HAT OPENSHIFT 上的 ANSIBLE AUTOMATION PLATFORM？

在 Red Hat OpenShift 4.x 上运行 Ansible Automation Platform 2.3 可提供更有效且自动化的方法来部署和管理平台，并更好地与 Red Hat OpenShift 的内置监控、日志记录、安全性和可扩展性功能集成。

另外，使用 Red Hat OpenShift 的 Operator Lifecycle Manager (OLM)来部署和管理 Ansible Automation Platform 操作器可以简化更新和升级过程，从而提高控制和灵活性。

对这些优点的进一步分类如下：

- 自动化 ： Ansible Automation Platform Operator 提供了在 Red Hat OpenShift 上自动部署和管理 Ansible Automation Platform 的方法。这有助于您减少部署和管理平台所需的时间和精力，并确保它以一致且可预测的方式运行。
- **可扩展性：** Operator 可帮助您扩展 Ansible Automation Platform，以满足您的机构需求。您可以轻松部署多个平台实例，并从单个位置进行管理。
- **flexibility**: Operator 提供了部署和管理 Ansible Automation Platform 的灵活方法。您可以自定义平台的配置以满足您的特定需求，并将其部署到不同的环境中，如开发、临时和生产环境中。
- **监控和故障排除：** 操作器与 Red Hat OpenShift 的内置监控和日志记录工具集成，如 Prometheus 和 Grafana，可用于监控平台资源使用情况并识别瓶颈。
- **管理和升级：** 使用附带 Red Hat OpenShift 4.x 的 Operator Lifecycle Manager (OLM)，您可以轻松地部署、管理和升级[1] 集群中的 Ansible Automation Platform Operator。

[1] Ansible Automation Platform 支持的生命周期版本 - https://access.redhat.com/support/policy/updates/ansible-automation-platform

第 3 章 开始前

在 Red Hat OpenShift 上部署 AAP 之前，务必要了解在安装前需要解决的关键注意事项。这些因素决定了您的 AAP 环境在整个生命周期中的健康状态和可扩展性。

在本节中，您将发现这些关键注意事项，包括：

- Red Hat OpenShift 资源管理
- 自动化控制器 pod 容器的大小建议
- Postgres pod 的大小建议
- 自动化作业 pod 的大小建议
- Automation hub pod 的大小建议

成功部署的一个关键方面是 Pod 和容器的正确资源管理，以确保 Red Hat OpenShift 集群的 AAP 应用程序的最佳性能和可用性。

3.1. POD 和容器的资源管理

有关资源管理的两个关键资源是您的 CPU 和内存(RAM)。Red Hat OpenShift 使用资源请求和资源限制来控制容器可在 pod 中消耗的资源量。

3.1.1. 什么是资源请求？

资源请求是容器正确运行和正常工作所需的最小资源量。Kubernetes 调度程序使用这个值来确保有足够的资源可供容器使用。

3.1.2. 什么是资源限值？

另一方面，资源限值是容器可以使用的最大资源量。设置资源限值可确保容器不会消耗的资源超过应该的资源，这可能会导致其他容器遭遇资源不足。

3.1.3. 为什么资源管理很重要？

当它指向 AAP 时，设置正确的资源请求和限值至关重要。资源分配可能会导致控制 pod 终止，从而导致自动化控制器中的所有自动化作业丢失。

3.1.4. 规划资源

在设置正确的资源管理值时，组织需要根据可用资源考虑最适合其需求的架构。例如，确定其 Ansible Automation Platform 环境的高可用性是否比最大化运行自动化作业的容量更重要。

为了更好地说明，让我们获得此参考架构中使用的现有 Red Hat OpenShift 环境。它由以下几项组成：

- 3 个 control plane 节点
- 3 个 worker 节点

这些节点由 4 个 vCPU 和 16 GiB RAM 组成。

由于 Red Hat OpenShift 集群的 control plane 节点没有运行任何应用程序，因此示例侧重于可用的 3 个 worker 节点。

对于这 3 个 worker 节点，我们需要确定更重要的信息：最大化 Ansible Automation Platform 的可用性或运行尽可能多的自动化作业？

如果可用性非常重要，则重点是确保两个控制 pod 在单独的 worker 节点上运行（如 **worker0** 和 **worker1**），而所有自动化作业都在剩余的 worker 节点上运行（如 **worker2**）。

但是，这可减少运行自动化作业的资源（以半计），因为推荐的做法是将控制 pod 和自动化 pod 分隔开来。

如果要最大化运行自动化作业的数量是主要目标，那么对控制 pod 使用一个 worker 节点（如 **worker0**）并使用剩余的两个 worker 节点（例如， **worker1 和 worker2**）来运行自动化作业会加倍可用的资源，但代价是 control pod 的冗余。

当然，解决方案可能同样重要，在这种情况下，额外的资源（例如添加更多 worker 节点）需要满足这两个要求。

3.2. 自动化控制器 POD 容器的大小建议

从概述部分查看 图 1.1 “自动化控制器架构”，您会注意到 control pod 包含 4 个容器：

- web
- ee
- redis
- task

这些容器在 Ansible 自动化控制器中执行一个唯一功能，了解资源配置如何影响控制 pod 至关重要。默认情况下，Red Hat OpenShift 提供了适用于最小测试安装的低值，但不建议在生产环境中运行 Ansible Automation Platform。

Ansible Automation Platform 的 Red Hat OpenShift 默认值是：

- cpu: 100m
- 内存：128Mi

默认情况下，Red Hat OpenShift 不会配置任何最大资源限值，并将尝试分配 Ansible Automation Platform 控制 pod 请求的所有可能资源。此配置可能会导致资源不足，并影响在 Red Hat OpenShift 集群上运行的其他应用程序。

要演示控制 Pod 中容器的资源请求和限值的起点，我将使用以下假设：

- 3 个 worker 节点在 Red Hat OpenShift 集群中可用，每个都有 4 个 vCPU 和 16GiB RAM
- 为自动化作业最大化资源比高可用性更重要
- 用于运行自动化控制器的专用 worker 节点
- 剩余的两个 worker 节点用于运行自动化作业

在调整控制 pod 中的容器大小时，务必要考虑具体工作负载。虽然我进行了性能测试，它们为这个参考环境提供特定建议，但这些建议可能不适用于所有类型的工作负载。

作为起点，我决定利用 性能集合 playbook，特别是 chatty_tasks.yml。

性能基准包括：

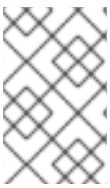
- 使用 1 个主机创建清单
- 创建运行 **chatty_tasks.yml** 文件的作业模板

chatty_tasks 作业模板使用 **ansible.builtin.debug** 模块来生成每个主机的一组调试消息，并生成必要的清单。通过使用 **ansible.builtin.debug** 模块，我可以在不引入任何其他开销的情况下获得自动化控制器性能的准确表示。

作业模板使用指定的并发级别执行，范围从 10 到 50，表示作业模板的同时调用数量。

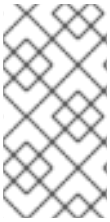
以下 **资源请求和资源限制** 是性能基准的结果，可用作使用类似资源的 Red Hat OpenShift 集群运行 AAP 的起始基准。

```
spec:
...
  ee_resource_requirements:
    limits:
      cpu: 500m
      memory: 400Mi
    requests:
      cpu: 100m
      memory: 400Mi
  task_resource_requirements:
    limits:
      cpu: 4000m
      memory: 8Gi
    requests:
      cpu: 1000m
      memory: 8Gi
  web_resource_requirements:
    limits:
      cpu: 2000m
      memory: 1.5Gi
    requests:
      cpu: 500m
      memory: 1.5Gi
  redis_resource_requirements:
    limits:
      cpu: 500m
      memory: 1.5Gi
    requests:
      cpu: 250m
      memory: 1.5Gi
```



注意

内存资源请求和限值匹配以防止在 Red Hat OpenShift 集群中利用内存资源，这可能会导致 Pod 的 Out Of Memory (OOM) Kill。如果资源限值大于资源请求，可能会导致一个场景，允许过度使用 Red Hat OpenShift 节点。



注意

CPU 资源请求和限值与内存不同，因为 CPU 资源被视为可压缩。这意味着，Red Hat OpenShift 会在达到资源限值时尝试节流容器的 CPU，但不会终止容器。在以上控制 pod 中的容器中，提供了 CPU 请求，这些请求对于给定的工作负载有足够的 CPU，但允许将其阈值(CPU 限值)设置为更高的值，允许在负载下激增。

警告

上面的场景假设，没有其他应用程序在 control pod 所在的 worker 节点上使用资源，因为它使用了专用的 Red Hat OpenShift worker 节点。更多详情请参阅 第 3.7 节 "为自动化控制器 pod 指定专用节点"。

3.3. POSTGRES POD 的大小建议

在使用 **chatty_task** playbook 执行性能基准测试后，观察到 500m 以下的 CPU 资源请求可能会导致 Postgres pod 中的 CPU 节流，因为初始资源请求上方请求的其他资源（在资源限值下）不能保证 pod。但是，CPU 限值被设置为 1000m (1 vCPU)，因为测试过程中会出现超过 500m 请求的突发。

对于内存，因为内存不是可压缩的资源，因此在 **chatty_task** 性能测试 Postgres pod 在测试中其最高级别的测试中略消耗了 650Mi RAM。

因此，根据结果，对于此参考环境，我的内存资源请求和限制推荐 1Gi 提供足够的缓冲区，并避免 Postgres pod 的潜在内存不足(OOM) Kill。

以下 **资源请求和资源限制** 是性能基准测试的结果，可用作运行 Postgres Pod 的起始基准。

```
spec:
...
  postgres_resource_requirements:
    limits:
      cpu: 1000m
      memory: 1Gi
    requests:
      cpu: 500m
      memory: 1Gi
```

警告

以下值特定于此参考环境，可能不适用于您的工作负载。务必要监控 Postgres pod 的性能，并调整资源分配以满足您的性能需求。

3.4. 自动化作业 POD 的大小建议

Ansible Automation Platform 作业是针对主机清单启动 Ansible playbook 的自动化控制器实例。当 Ansible Automation Platform 在 Red Hat OpenShift 上运行时，默认执行队列是 Operator 在安装时创建的容器组。

容器组由 Kubernetes 凭证和默认 Pod 规格组成。当作业启动到容器组时，由容器组 pod 规格指定的命名空间中的自动化控制器创建 pod。这些 pod 被称为自动化作业 pod。

要确定用于自动化作业 pod 的大小，必须首先了解自动化控制器 control plane 可同时启动的作业数量。

在本例中，有 3 个 worker 节点（每个 4 个 vCPU 和 16GiB RAM）。一个 worker 节点托管 control pod，另一个两个 worker 节点用于自动化作业。

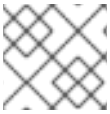
根据这些值，我们可以确定自动化控制器 control plane 可以运行的控制容量。

以下公式提供分类：

control capacity = Total Memory in MB / Fork size in MB

基于 worker 节点，这可表达为：

总控制容量 = 16,000 MB / 100 MB = 160



注意

对于对计算的更多详细信息感兴趣的证书，请查看资源确定容量算法

这意味着自动化控制器被配置为同时启动 160 个作业。但是，需要对这个值进行调整才能与我们的容器组/execution plane 容量相匹配。

注意

为简单起见，16GB 被舍入为 16,000 MB，一个分叉的大小默认为 100MB。

现在，我们计算了可用的控制容量，我们就可以确定并发自动化作业的最大数量。

要确定这一点，我们必须注意，容器组/execution plane 中的自动化作业 pod 规格具有 vCPU 250m 和 100Mi RAM 的默认请求。

使用一个 worker 节点的总内存：

16,000 MB / 100 MiB = 160 个并发作业

使用一个 worker 节点的总 CPU：

4000 millicpu / 250 millicpu = 16 个并发作业

根据以上值，我们必须将节点上的最大并发作业设置为两个并发作业值的最小 - 16。因为有两个 worker 节点分配给在我们的示例中运行自动化作业，这个数字会加倍到 32 （每个 worker 节点有 16 个并发作业）。

自动化控制器的配置目前设置为 160 个并发作业，可用的 worker 节点容量只允许 32 个并发作业。这是一个问题，因为数字没有平衡。

这意味着自动化控制器的 control plane 认为它可以同时启动 160 个作业，而 Kubernetes 调度程序只会在 Container Group 命名空间中同时调度最多 32 个自动化作业 pod。

control plane 和容器组/execution plane 之间的未平衡值可能会导致问题：

- 如果 control plane 容量高于 Container Group 的最大并发作业 pod 数量，则 control plane 将尝试通过提交 pod 启动来启动作业。但是，这些 pod 不会实际开始运行，直到资源可用为止。如果作业 pod 在 **AWX_CONTAINER_GROUP_POD_PENDING_TIMEOUT** 的超时时间内启动，则该作业将中止（默认为 2 小时）。

- 如果容器组能够支持比 control plane 可以启动更多的并发自动化作业，则此容量将实际发生，因为自动化控制器将无法启动足够的自动化作业，以便访问容器组可以支持的最大并发自动化作业。

为了避免中止的作业或未使用资源的风险，建议把有效的控制容量与默认容器组支持的最大并发作业数量相平衡。

使用术语"有效控制容量"，因为 control plane 将启动的最大作业数量受名为 **AWX_CONTROL_NODE_TASK_IMPACT** 的设置的影响。**AWX_CONTROL_NODE_TASK_IMPACT** 变量定义了每个自动化作业可在控制 pod 上消耗的容量量，从而有效地控制控制 pod 将启动的自动化作业数量。

要在有效控制容量和可用执行容量之间实现平衡，我们可以将 **AWX_CONTROL_NODE_TASK_IMPACT** 变量设为限制要在自动化控制器 control plane 上运行的并发作业数量的值。

要计算 **AWX_CONTROL_NODE_TASK_IMPACT** 的最佳值，以避免启动比容器组支持的更并发自动化作业，我们可使用以下公式：

AWX_CONTROL_NODE_TASK_IMPACT = 控制容量/容器组可以启动的最大并发作业

对于我们的参考环境，这是：

AWX_CONTROL_NODE_TASK_IMPACT = 160 / 32 = 5

本参考环境的此结束，XX **_CONTROL_NODE_TASK_IMPACT** 应该等于 5。此值将在 第 6 章 *安装自动化控制器* 章节的 **extra_setting** 部分中设置，我们将在本文档的后面部分介绍。

3.5. 自动化控制器 POD 大小建议概述

正确设置 control plane (control pod)和容器组/execution plane （自动化作业 pod）的资源请求和限值，以确保控制和执行容量是均衡的。正确的配置可由以下决定：

- 计算控制容量
- 计算可以同时运行的自动化作业数量
- 使用安装自动化控制器中的适当平衡值设置 **AWX_CONTROL_NODE_TASK_IMPACT** 变量

3.6. AUTOMATION HUB POD 的大小建议

在概述部分中所述的 图 1.2 “Automation hub 架构” 中，您会注意到部署由 7 个 pod 组成，每个 pod 托管一个容器。

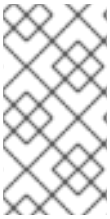
pod 列表包括：

- 内容(x2)
- redis

- api
- postgres
- worker (x2)

组成自动化中心架构的七种 pod 可以一起工作，以高效管理和分发内容，并且对自动化中心环境的整体性能和可扩展性至关重要。

在这些 pod 中，worker pod 特别重要，因为它们负责处理、同步和分发内容。因此，务必要将适当的资源设置为 worker pod，以确保它们能够执行任务。



注意

以下指南旨在提供自动化中心环境所需的资源请求和限值的估算。实际资源需求将因设置而异。

例如，在执行更新或同步的大量存储库的环境可能需要更多资源来处理处理负载。

在这个参考环境中，为了确定 pod 的大小，预先精简测试是使用在自动化中心环境中进行的最高内存消耗任务之一来实现的。

该发现确定，在自动化中心中成功同步远程仓库，需要为每个 pod 设置以下资源请求和资源限制：

```
spec:
...
content:
  resource_requirements:
    limits:
      cpu: 250mm
      memory: 400Mi
    requests:
      cpu: 100m
      memory: 400Mi

redis:
  resource_requirements:
    limits:
      cpu: 250m
      memory: 200Mi
    requests:
      cpu: 100m
      memory: 200Mi

api:
  resource_requirements:
    limits:
      cpu: 250m
      memory: 400Mi
    requests:
      cpu: 150m
      memory: 400Mi

postgres_resource_requirements:
  resource_requirements:
```

```
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 1Gi

worker:
  resource_requirements:
    limits:
      cpu: 1000m
      memory: 3Gi
    requests:
      cpu: 400m
      memory: 3Gi
```

3.7. 为自动化控制器 POD 指定专用节点

在专用节点上运行控制 pod 非常重要，以便分隔控制 pod 和自动化作业 pod，并防止这两种类型的 pod 间的资源争用。这种分离有助于维护控制 pod 及其提供的服务的稳定性和可靠性，而不会因为资源限制而造成降级风险。

在此参考环境中，重点是最大程度提高可以运行的自动化作业的数量。这意味着，在 Red Hat OpenShift 环境中可用的 3 个 worker 节点，一个 worker 节点专用于运行 control pod，而其他 2 个 worker 节点则用于执行自动化作业。

警告

如果专用 worker 节点要停机，则只使用一个 worker 节点来运行 control pod 的潜在风险，因为当专用 worker 节点要停机时，该服务不会启动。要补救这种情况，请减少运行自动化作业的 worker 节点数量，或添加额外的 worker 节点，以便在 Red Hat OpenShift 集群中运行额外的控制 pod 副本是可行的选项。

3.7.1. 为自动化控制器将控制 pod 分配给特定的 worker 节点

要将 control pod 分配给 Red Hat OpenShift 中的特定节点，使用 pod 规格中的 **node_selector** 字段以及 **topology_spread_constraints** 字段的组合。**node_selector** 字段允许您指定节点必须匹配的标签条件，才能有资格托管该 pod。例如，如果您有一个带有标签 **aap_node_type: control** 的节点，在 pod 规格中指定以下内容以将 pod 分配给此节点：

```
spec:
...
  node_selector: |
    aap_node_type: control
```

topology_spread_constraints 将可在标签 **aap_node_type: control** 设置为 1 的节点上的 pod (**maxSkew**)的最大数量。**topologyKey** 设置为 **kubernetes.io/hostname**，它是指示节点主机名的内置标签。**whenUnsatisfiable** 设置被设置为 **ScheduleAnyway**，允许在没有足够的节点满足约束时调度

pod。**labelSelector** 与带有标签 **aap_node_type: control** 的 pod 匹配。这样做的影响是 Red Hat OpenShift 优先选择每个节点调度单个控制器 pod。但是，如果存在比可用的 worker 节点更多的副本请求，如果有足够的资源可用，Red Hat OpenShift 允许在同一 worker 节点上调度多个控制器 pod。

tolerations 部分指定 pod 只能调度到带有标签 **dedicated: AutomationController** 的节点上，容限的效果被设为 **NoSchedule**，确保 pod 不会调度到没有所需标签的节点。这用于与 **topology_spread_contstraints** 结合使用，不仅指定如何在节点之间分散 pod，还用于指示它们可以调度到哪些节点。

```
spec:
...
  topology_spread_constraints: |
    - maxSkew: 1
      topologyKey: "kubernetes.io/hostname"
      whenUnsatisfiable: "ScheduleAnyway"
      labelSelector:
        matchLabels:
          aap_node_type: control
  tolerations: |
    - key: "dedicated"
      operator: "Equal"
      value: "AutomationController"
      effect: "NoSchedule"
```

注意

节点标签和污点的应用程序可在 附录 C, *将标签和污点应用到 Red Hat OpenShift 节点* 中找到。为 spec 文件添加节点选择器、拓扑约束和容限的步骤在 第 6 章 *安装自动化控制器* 中显示。

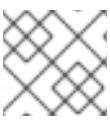
3.8. 处理数据库高可用性

在 Ansible Automation Platform 中部署自动化控制器和自动化中心组件，可以利用其 PostgreSQL 数据库的 PVC。确保这些 PVC 的可用性对于运行 Ansible Automation Platform 的稳定性至关重要。

有几个策略可用于处理 Red Hat OpenShift 集群中的 PVC 可用性，如 Crunchy Data 通过 Postgres Operator (PGO)和 OpenShift Data Foundation (ODF)提供的 PVC 的可用性。

Crunchy Data 提供 PGO (Postgres Operator)，它为您提供了一个声明性 Postgres 解决方案，用于自动管理 PostgreSQL 集群。通过 PGO，用户可以创建其 Postgres 集群，扩展并创建高可用性(HA) Postgres 集群，并将其连接到 Ansible Automation Platform 等应用程序。

OpenShift Data Foundation (ODF)是一个高度可用的存储解决方案，可以管理容器化应用程序的持久性存储。它由多种开源操作器和技术组成，包括 Ceph、NooBaa 和 Rook。这些不同的 Operator 允许您置备和管理文件、块和对象存储，然后可以连接到 Ansible Automation Platform 等应用程序。



注意

为 PostgreSQL 数据库提供高可用性 PVC 的步骤超出了此参考架构的范围。

第 4 章 先决条件

此参考环境的 Ansible Automation Platform 2.3 安装使用以下内容：

- Red Hat OpenShift Platform 4.12
- 附录 C, *将标签和污点应用到 Red Hat OpenShift 节点*
- 用于处理自动化中心 **ReadWriteMany** 存储要求的 Amazon S3 存储桶

警告

安装 AAP 2.3 **需要至少** 版本 Red Hat OpenShift 4.9。如需了解更多详细信息，请访问 Red Hat Ansible Automation Platform 生命周期页面 。

注意

部署 Red Hat OpenShift 的方法有很多。Red Hat OpenShift 集群的大小将取决于您的应用程序的具体要求（不仅仅是 Ansible Automation Platform）。要考虑的因素包括访问集群的用户数量、应用程序需要运行的数据和资源数量，以及您的可扩展性和冗余要求。

有关如何部署 Red Hat OpenShift 的详细信息，请参阅安装指南

除了以上安装指南外，请参阅 OpenShift 4 资源配置：方法和工具 博客文章，以帮助根据您的需要确定适当的集群大小。

注意

自动化 hub 需要基于 **ReadWriteMany** 文件、Azure Blob 存储或 Amazon S3 兼容存储才能进行操作，以便多个 pod 可以访问共享内容，如集合。

此参考环境利用创建 Amazon S3 存储桶来满足 **ReadWriteMany** 存储的请求。详情 附录 D, *创建 Amazon S3 存储桶*。

第 5 章 安装 ANSIBLE AUTOMATION PLATFORM OPERATOR

安装 Ansible Automation Platform Operator 时，首选部署方法是，在带有手动更新批准的目标命名空间中安装集群范围的 Operator。

此部署方法的主要优点是，它只影响目标命名空间中的资源，这些资源在希望限制 AAP Operator 的范围时，可以在不同命名空间中处理它时提供灵活性。

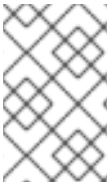
例如，如果您打算在测试升级时拥有单独的 *devel* 和 *prod* 命名空间来管理不同的 AAP 部署。

部署 Ansible Automation Platform Operator 的步骤如下。

- 使用集群凭证登录到 Red Hat OpenShift Web 控制台。
- 在左侧导航菜单中选择 Operators → OperatorHub。
- 搜索 Ansible Automation Platform 并选择它。
- 在 Ansible Automation Platform Install 页面中，选择 "Install"。
- 在"Install Operator"页面中，
  - 选择适当的更新频道 **stable-2.3-cluster-scoped**
  - 选择适当的安装模式 **A specific namespace on the cluster**
  - 选择适当的安装的命名空间，**Operator 建议命名空间： aap**
  - 选择适当的更新批准，例如 **Manual**。
- 点 **Install**。
- 点 Manual approval 所需的 **Approve**。

安装 Ansible Automation Platform 的过程可能需要几分钟时间。

安装完成后，选择 **View Operator** 按钮来查看安装过程中指定的命名空间中已安装的 Operator （如 **aap**）。



注意

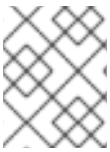
这个 AAP operator 部署只以命名空间 **aap** 为目标。如果 AAP 操作器要作为目标（管理）其他命名空间，则必须将它们添加到 **OperatorGroup spec** 文件中。详情 附录 F, *在 AAP Operator 中添加额外的受管命名空间*。

注意

Ansible Automation Platform Operator 的默认资源值适合典型的安装。但是，如果部署大量自动化控制器和自动化中心环境，建议使用 **subscription.spec.config.resources** 在订阅 spec 中增加 Ansible Automation Platform Operator 的资源阈值。这样可确保 Operator 有足够的资源来处理增加的工作负载并防止性能问题。

第 6 章 安装自动化控制器

安装 Ansible Automation Platform Operator 后，以下步骤在 Red Hat OpenShift 集群中安装自动化控制器。



注意

资源请求和限值值特定于此引用环境。确保阅读 第 3 章 *开始前* 部分来正确地计算 Red Hat OpenShift 环境的值。

警告

删除自动化控制器实例时，关联的持久性卷声明(PVC)不会被自动删除。如果新部署的名称与之前部署的名称相同，这可能会导致迁移期间出现问题。建议您在在同一命名空间中部署新自动化控制器实例前删除旧的 PVC。删除以前部署 PVC 的步骤可在 附录 B, *从以前的 AAP 安装中删除现有的 PVC* 中找到。

- 使用集群凭证登录到 Red Hat OpenShift Web 控制台。
- 在左侧导航菜单中选择 **Operators → Installed Operators**，选择 **Ansible Automation Platform**。
- 进入到 **Automation Controller** 选项卡，然后点 **Create AutomationController**。
- 在 Form 视图中，提供 **Name**，例如 *my-automation-controller* 并选择 **高级配置** 来扩展附加选项。
- 在 **Additional configuration** 中，按照从启动前部分计算的每个容器设置适当的 **资源要求**。
  - **扩展 Web Container 资源要求**
    - **limits: CPU 内核：2000m, Memory: 1.5Gi**
    - **requests: CPU cores: 500m, Memory: 1.5Gi**
  - **扩展 任务容器资源要求**
    - **limits: CPU cores: 4000m, Memory: 8Gi**
    - **requests: CPU cores: 1000m, Memory: 8Gi**
  - **展开 EE Control Plane Container 资源要求**
    - **limits: CPU cores: 500m, Memory: 400Mi**
    - **requests: CPU cores: 100m, Memory: 400Mi**
  - **扩展 Redis Container 资源要求**
    - **limits: CPU cores: 500m, Memory: 1.5Gi**
    - **requests: CPU cores: 250m, Memory: 1.5Gi**

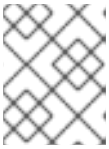
- 扩展 PostgreSQL Container 资源要求
  - limits: CPU 内核：1000m, Memory: 1Gi
  - requests: CPU cores: 500m, Memory: 1Gi

- 在 Create AutomationController 页面的顶部，切换 YAML 视图
  - 在 **spec:** 部分中，添加 **extra_settings** 参数，以传递 第 3 章 *开始前* 部分中计算的 **AWX_CONTROL_NODE_TASK_IMPACT** 值

```
spec:
...
  extra_settings:
    - setting: AWX_CONTROL_NODE_TASK_IMPACT
      value: "5"
```

- 在 YAML 视图 中，将以下内容添加到 spec 部分，为 control pod 添加专用节点。

```
spec:
...
  node_selector: |
    aap_node_type: control
  topology_spread_constraints: |
    - maxSkew: 1
      topologyKey: "kubernetes.io/hostname"
      whenUnsatisfiable: "ScheduleAnyway"
      labelSelector:
        matchLabels:
          aap_node_type: control
  tolerations: |
    - key: "dedicated"
      operator: "Equal"
      value: "AutomationController"
      effect: "NoSchedule"
```



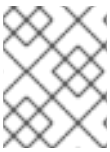
注意

确保您的节点标签和污点到应运行控制 pod 的适当专用 worker 节点。要设置的详情可在 附录 C, *将标签和污点应用到 Red Hat OpenShift 节点*中找到。

- 点 Create 按钮

第 7 章 安装自动化中心

安装 Ansible Automation Platform Operator 后，以下步骤在 Red Hat OpenShift 集群中安装自动化中心。

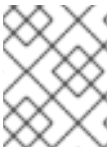


注意

资源请求和限值值特定于此引用环境。确保阅读 第 3 章 *开始前* 部分来正确地计算 Red Hat OpenShift 环境的值。

警告

删除自动化中心实例时，关联的持久性卷声明(PVC)不会被自动删除。如果新部署的名称与之前部署的名称相同，这可能会导致迁移期间出现问题。建议您在在同一命名空间中部署新自动化中心实例前删除旧的 PVC。删除以前部署 PVC 的步骤可在 附录 B, *从以前的 AAP 安装中删除现有的 PVC* 中找到。



注意

自动化 hub 需要基于 **ReadWriteMany** 文件的存储、Azure Blob 存储或 Amazon S3 兼容存储才能确保多个 pod 可以访问共享内容，如集合。

- 使用集群凭证登录到 Red Hat OpenShift Web 控制台。
- 在左侧导航菜单中选择 Operators → Installed Operators，选择 Ansible Automation Platform。
- 进入 Automation Hub 选项卡，然后点 Create AutomationHub。
- 在 Form 视图中
  - 提供一个名称， 如 *my-automation-hub*
  - 在 Storage type 中，选择您的 **ReadWriteMany** 兼容存储。

    注意

    此引用环境使用 Amazon S3 作为其 **ReadWriteMany** 存储。有关创建 Amazon S3 存储桶的详情，请参考 附录 D, *创建 Amazon S3 存储桶*。

    - 提供 S3 存储机密。有关如何在 附录 E, *创建 AWS S3 Secret* 中创建的详情。
  - 选择 Advanced configuration 以展开附加选项。
  - 在 PostgreSQL 容器存储要求中（使用受管实例时）
    - 将存储限制设置为 50Gi
    - 将存储请求设置为 8Gi

- 在 PostgreSQL 容器资源要求中（使用受管实例时）
  - limits: CPU cores: 500m, Memory: 1Gi
  - requests: CPU cores: 200m, Memory: 1Gi
- 在 Redis 部署配置中，选择 Advanced configuration
  - 选择内存中数据存储资源要求
    - limits: CPU cores: 250m, Memory: 200Mi
    - requests: CPU cores: 100m, Memory: 200Mi
- 在 API 服务器配置 中，选择 Advanced configuration
  - 选择 API 服务器资源要求
    - limits: CPU cores: 250m, Memory: 400Mi
    - requests: CPU cores: 150m, Memory: 400Mi
- 在 Content server configuration 中，选择 Advanced configuration
  - 选择 Content server 资源要求
    - limits: CPU cores: 250m, Memory: 400Mi
    - requests: CPU cores: 100m, Memory: 400Mi
- 在 Worker 配置中，选择 Advanced configuration
  - 选择 Worker 资源要求
    - limits: CPU 内核：1000m, Memory: 3Gi
    - requests: CPU cores: 500m, Memory: 3Gi

- 点 Create 按钮

第 8 章 登录到您的自动化控制器仪表板

成功安装自动化控制器后，可以通过以下步骤访问仪表板：

- 在 Red Hat OpenShift Web 控制台中，在左侧导航菜单中选择 Operators→ Installed Operators。
- 选择 Ansible Automation Platform。
- 在 Operator Details 中，选择 Automation Controller 选项卡。
- 选择安装的自动化控制器的 Name。
- 在自动化控制器 概述中，提供了包括 URL、Admin 用户和 Admin 密码的详细信息。

第 9 章 登录到您的自动化中心仪表板

- 在 Red Hat OpenShift Web 控制台中，在左侧导航菜单中选择 Operators→ Installed Operators。
- 选择 Ansible Automation Platform。
- 在 Operator Details 中，选择 Automation Hub 选项卡。
- 选择已安装的自动化中心 的名称。
- 在 Automation Hub 概述 中，提供了包括 URL、Admin 用户和 Admin 密码的详细信息。

第 10 章 监控 ANSIBLE AUTOMATION PLATFORM

成功安装 Ansible Automation Platform 后，优先选择维护其健康状况和监控关键指标的能力至关重要。

本节重点介绍如何监控由 Red Hat OpenShift 中新安装的 Ansible Automation Platform 环境提供的 API 指标。

10.1. 哪些将用于监控 API 指标？

Prometheus 和 Grafana。

Prometheus 是一个用于收集和聚合指标的开源监控解决方案。通过 Grafana 合作伙伴 Prometheus 监控功能，这是运行数据分析的开源解决方案，并在可自定义仪表板中拉取指标，并实时视觉化指标来跟踪 Ansible Automation Platform 的状态和健康状况。

10.2. 我应该查看什么指标？

Grafana 预建仪表板显示：

- Ansible Automation Platform 版本
- 控制器节点数量
- 许可证中可用的主机数
- 使用的主机数量
- 用户总数
- 任务成功
- 作业失败
- 按作业执行类型的数量
- 显示运行和待处理的作业数量的图形
- 图显示工具的增长，显示工作流、主机、清单、作业、项目、机构等的数量。

可以自定义此 Grafana 仪表板，以捕获您可能感兴趣的其他指标。但是，自定义 Grafana 仪表板超出了此参考架构的范围。

10.3. 通过 ANSIBLE PLAYBOOK 安装

使用自定义 Grafana 仪表板通过 Prometheus 监控 Ansible Automation Platform 的过程可在几分钟内安装。以下提供了利用预构建的 Ansible playbook 的步骤。

成功运行 Ansible Playbook 需要以下步骤：

- 在自动化控制器中创建自定义凭证类型
- 在自动化控制器中创建 kubeconfig 凭证
- 创建项目和作业模板以运行 Ansible Playbook

10.3.1. 创建自定义凭证类型

在 Ansible Automation Platform 仪表板中，

1. **在 Administration→Credential Types 下，点蓝色 添加按钮。**
2. **提供一个名称，如 *Kubeconfig***
3. **在输入配置中输入以下 YAML：**

```
fields:
  - id: kube_config
    type: string
    label: kubeconfig
    secret: true
    multiline: true
```

4. **在注入器配置中输入以下 YAML：**

```
env:
  K8S_AUTH_KUBECONFIG: '{{ tower.filename.kubeconfig }}'
file:
  template.kubeconfig: '{{ kube_config }}'
```

5. **点击 Save。**

10.3.2. 创建 kubeconfig 凭证

在 Ansible Automation Platform 仪表板中，

1. **在 Resources→Credentials 下，点蓝色 添加按钮。**
2. **提供一个名称， 如 *OpenShift-Kubeconfig***
3. **在 凭证类型下拉菜单中，选择 Kubeconfig。**
4. **在 Type Details 文本框中，为您的 Red Hat OpenShift 集群插入 kubeconfig 文件**
5. **点击 Save。**

10.3.3. 创建一个项目

在 Ansible Automation Platform 仪表板中，

1. **在 Resources→Projects 下，点蓝色 添加按钮。**
2. **提供一个名称， 例如 *监控 AAP 项目***
3. **选择 Default 作为机构。**
4. **选择 Default execution environment 作为 Execution Environment。**
5. **选择 Git 作为 Source Control Credential Type。**
6. **在 类型详情中，**

a. 添加 Source Control URL (https://github.com/ansible/aap_ocp_refarch)

7. 在 选项 中，
   a. 选择 Clean, Delete, Update Revision on Launch。
8. 点击 Save。

10.3.4. 创建作业模板并运行 Ansible Playbook

在 Ansible Automation Platform dsahboard 中，

1. 在 Resources→Templates 下点 blueAdd→Add 作业模板
2. 提供一个名称，如监控 *AAP 作业*
3. 选择 Run 作为 Job Type。
4. 选择 Demo Inventory 作为 清单。
5. 选择 Monitoring AAP Project 作为项目。
6. 选择 Default execution environment 作为 Execution Environment。
7. 选择 *aap-prometheus-grafana/playbook.yml* 作为 Playbook。
8. 选择 Credentials，并将类别从 Machine 切换到 Kubeconfig。
9. 选择适当的 *kubeconfig* 以访问 Red Hat OpenShift 集群，如*OpenShift-Kubeconfig*
10. 可选步骤 ：在变量中，可以修改以下变量：
    a. prometheus_namespace: <*your-specified-value*>
    b. ansible_namespace: <*your-specified-value*>
11. 点击 Save。
12. 点 Launch 运行 Ansible Playbook
13. 作业输出中会显示登录到 Grafana 和 Prometheus 的详情

附录 A. 关于 AUTHOR

Roger Lopez

Roger Lopez is a Principal Technical Marketing Manager bringing 10+ years of computer industry experience delivering high-value solutions used by our sales, marketing and engineering teams to develop best practice documentation & methods for internal and external customers. He is a Red Hat Certified Engineer (RHCE) with experience building solutions around Ansible, OpenShift and OpenStack.

附录 B. 从以前的 AAP 安装中删除现有的 PVC

1. **打开终端窗口并登录到 Red Hat OpenShift 集群，例如导出 KUBECONFIG 文件**

```
$ export KUBECONFIG=/path/to/kubeconfig
```

2. **验证指定 Ansible Automation Platform 命名空间中的 PVC 列表，例如 aap**

```
$ oc get pvc -n aap
```

注意

这显示了 aap 命名空间中的所有 PVC 列表，包括其名称、状态、容量、访问模式和存储类。

3. **识别您要从列表中删除的 PVC**

4. **使用 oc delete 命令删除 PVC**

```
oc delete pvc <pvc-name-to-remove> -n aap
```

验证 aap 命名空间中可用的 PVC 列表，并确保 PVC 不再显示。

```
$ oc get pvc -n aap
```

附录 C. 将标签和污点应用到 RED HAT OPENSHIFT 节点

要让我们的控制 pod 在专用的 Red Hat OpenShift 节点上运行，必须将适当的标签和污点设置为指定节点。

在本例中，我们将选择具有角色 worker 的 Red Hat OpenShift 节点，标签为 aap_node_type=control。

1. **获取您要标记运行的节点的名称**

```
$ oc get nodes
```

2. **从列表中选择节点并记录其名称，如 worker1**

3. **将 aap_node_type=control 标签应用到节点**

```
$ oc label node <node-name> aap_node_type=control
```

注意

将 <node-name > 替换为您要标记的节点的名称。

4. **验证标签的创建，如下所示：**

```
$ oc get nodes --show-labels | grep <node-name>
```

创建标签后，下一步是将 NoSchedule 污点添加到已为其创建了标签的 worker 节点。

以下命令为节点添加一个 NoSchedule 污点：

```
oc adm taint nodes <node-name> dedicated=AutomationController:NoSchedule
```

专用 ：这是污点的键，它是提供的任意字符串，用于标识污点。

Automationcontroller: 这是为污点赋予的任意值。

NoSchedule: 这是污点的影响，它没有指定不容许此污点的 pod 将调度到此节点上。

通过将此污点应用到节点，我们告知 Kubernetes 调度程序为容许该污点的特定类型工作负载保留此节点。在这种情况下，我们使用 dedicated=AutomationController 容限为工作负载保留节点。

5. **验证污点是否已应用**

```
$ oc get nodes \
-o jsonpath='{range.items[*]}{@.metadata.name}{"\t"}{@.spec.taints[*].key}:
{@.spec.taints[*].value}{"\n"}{end}' \
| grep AutomationController
```

附录 D. 创建 AMAZON S3 存储桶

1. **打开一个终端，并确保使用 AWS 凭证安装和配置 AWS CLI。**
2. **运行以下命令以创建新 S3 存储桶：**

```
$ aws s3 mb s3://<bucket-name> --region <region-name>
```

警告

bucket 名称必须是唯一名称。

3. **运行以下命令，以检查存储桶是否已成功创建**

```
$ aws s3 ls | grep <bucket-name>
```

附录 E. 创建 AWS S3 SECRET

在 Red Hat OpenShift 中，可以存储 secret 并使用它们与 Amazon S3 等外部服务进行身份验证。此引用环境利用 Amazon S3 存储桶来满足 **ReadWriteMany** 运行自动化中心的要求。

以下步骤演示了如何在 Red Hat OpenShift 集群中创建在 *第 7 章 安装自动化中心* 章节中使用的 secret。

```
$ cat s3-secret.yml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: s3-secret
  namespace: aap
stringData:
  s3-access-key-id: my_key
  s3-secret-access-key: my_access_key
  s3-bucket-name: my_bucket
  s3-region: my_region
```

```
$ oc create -f s3-secret.yml
```

附录 F. 在 AAP OPERATOR 中添加额外的受管命名空间

以下是将额外的受管命名空间添加到位于 aap 命名空间中的已存在的 AAP operator 的步骤。

- **登录到 Red Hat OpenShift UI。**
- **在 Operators→Installed Operators 中，选择 Ansible Automation Platform**
- **在 Details 页面中，向下滚动到 ClusterServiceVersion Details**
  - **在右列中点 OperatorGroup**
- **在 OperatorGroup 详细信息中，选择 YAML**
  - **在 spec 部分下添加任何其他命名空间，如 aap-devel**

```
spec:
...
  targetNamespaces:
    - aap
    - aap-devel
```

- **点 Save**

注意

在添加到 OperatorGroup spec 文件前，目标命名空间必须已经存在。

附录 G. 参考

- 每个人都应该了解的 Kubernetes 内存限值、OOMKilled pod 和 pizza 方
- Pulp 项目硬件要求
- 使用基于 Operator 的安装的 Red Hat Ansible Automation Platform 性能注意事项
- 在 OpenShift Container Platform 上部署 Red Hat Ansible Automation Platform Operator
- Pulp Operator 存储配置
- AWX Operator
- 闲置 PostgreSQL 连接消耗的资源
- 使用 OperatorGroup 的 Operator 限定范围
- Ansible Tower 和 Grafana Dashboards

附录 H. REVISION HISTORY

修订 1.0-0	2023-03-07	Roger Lopez

- Initial Release