Red Hat Ansible Automation Platform 2.3

Red Hat Ansible Automation Platform 自动化网格指南

本指南提供用于部署自动化网格的信息，作为 Ansible Automation Platform 环境的一部分

Last Updated: 2024-06-18

Red Hat Ansible Automation Platform 2.3 Red Hat Ansible Automation Platform 自动化网格指南

本指南提供用于部署自动化网格的信息，作为 Ansible Automation Platform 环境的一部分

法律通告

摘要

提供反馈： 如果您对本文档有任何改进建议，或发现错误，请联系技术支持，使用 Docs组件在 Ansible Automation PlatformJIRA 项目中创建一个问题。

目录

前言

感谢您对 Red Hat Ansible Automation Platform 的关注。Ansible Automation Platform 是一个商业产品，它可以帮助团队通过增加控制、知识、协调基于 Ansible 的环境来更好地管理多阶的复杂部署环境。

本指南帮助您了解安装 Ansible Automation Platform 后的安装要求和流程。本文档已更新，以包含 Ansible Automation Platform 最新版本的信息。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 计划 RED HAT ANSIBLE AUTOMATION PLATFORM 环境中的自动化网格

以下主题包含有助于在 Ansible Automation Platform 环境中规划自动化网格部署的信息。以下小节解释了组成自动化网格的概念，以及有关如何设计自动化网格拓扑的示例。包括了简单易用的拓扑示例来说明您可以部署自动化网格的各种方法。

1.1. 关于自动化网格

自动化网格是一个覆盖网络，旨在通过使用现有网络相互建立对等连接的节点，简化在大型且分散的 worker 集合中的工作分布。

Red Hat Ansible Automation Platform 2 使用自动化控制器和自动化中心替换 Ansible Tower 和隔离节点。自动化控制器通过其 UI、Restful API、RBAC、工作流和 CI/CD 集成提供控制平面，而自动化网格则可用于设置、发现、更改或修改组成控制和执行层的节点。

Automation mesh 引入了：

- 动态集群容量可独立扩展，允许您以最少的停机时间创建、注册、分组、分组和取消注册节点。
- 控制和执行平面分离，可让您独立于 control plane 容量来缩放 playbook 执行容量。
- 对延迟具有弹性的部署选择，可在不中断的情况下重新配置，并在出现中断时动态重新路由来选择不同的路径。网格路由更改。
- 包括符合联邦信息处理标准(FIPS)的双向多跃网格通信可能性的连接。

1.2. 控制和执行平面

自动化网格利用唯一的节点类型来创建 **control** 和 **execution** plane。在设计自动化网格拓扑前，了解更多有关控制和执行平面及其节点类型的信息。

1.2.1. Control plane（控制平面）

control plane 由混合和控制节点组成。除了项目更新和管理作业外，控制平面中的实例还运行持久的自动化控制器服务，如 Web 服务器和任务分配程序。

- **混合节点** - 这是 control plane 节点的默认节点类型，负责项目更新、管理作业和 **ansible-runner** 任务操作等自动化控制器运行时功能。混合节点也用于自动化执行。
- **控制节点** - 控制节点运行项目和清单更新和系统作业，但不能控制常规作业。这些节点上禁用了执行功能。

1.2.2. 执行平面

执行平面由代表 control plane 执行自动化且没有控制功能执行的节点组成。hop 节点用于通信。 **执行平面**中的节点仅运行用户空间作业，且可能在地理上将延迟较高的节点与控制平面分隔开。

- **执行节点** - 执行节点在 **ansible-runner** 下以 **podman** 隔离运行作业。此节点类型与隔离的节点类似。这是 execution plane 节点的默认节点类型。
- **hop 节点** - 类似于跳过主机，hop 节点会将流量路由到其他执行节点。hop 节点无法执行自动化。

1.2.3. Peers

对等关系定义节点到节点的连接。您可以在 **[automationcontroller]** 和 **[execution_nodes]** 组内定义对等点，或者使用 **[automationcontroller:vars]** 或 **[execution_nodes:vars]** 组

1.2.4. 定义自动化网格节点类型

本节中的示例演示了如何为清单文件中的主机设置节点类型。

您可以为 control plane 或 execution plane 清单组中的单节点设置 **node_type**。要为整个节点组定义节点类型，请在组的 **vars** 小节中设置 **node_type**。

- control plane **[automationcontroller]** 组中的 **node_type** 允许的值是 **hybrid** (默认) 和 **control**。
- **[execution_nodes]** 组中的 **node_type** 允许的值是 **execution** (默认) 和 **hop**。

混合节点

以下清单由 control plane 中的单个混合节点组成：

```
[automationcontroller]
control-plane-1.example.com
```

控制节点

以下清单由 control plane 中的单个控制节点组成：

```
[automationcontroller]
control-plane-1.example.com node_type=control
```

如果您在 control plane 节点的 **vars** 小节中将 **node_type** 设置为 **control**，则 control plane 中的所有节点都是控制节点。

```
[automationcontroller]
control-plane-1.example.com

[automationcontroller:vars]
node_type=control
```

执行节点

以下片段在 execution plane 中定义一个执行节点：

```
[execution_nodes]
execution-plane-1.example.com
```

hop 节点

以下片段在 execution plane 中定义一个跃点节点和一个执行节点。为每个单独节点设置 **node_type** 变量。

```
[execution_nodes]
execution-plane-1.example.com node_type=hop
execution-plane-2.example.com
```

如果要在组级别上设置 **node-type**，您必须为执行节点和跃点节点创建单独的组。

```
[execution_nodes]
execution-plane-1.example.com
execution-plane-2.example.com

[execution_group]
execution-plane-2.example.com

[execution_group:vars]
node_type=execution

[hop_group]
execution-plane-1.example.com

[hop_group:vars]
node_type=hop
```

对等连接

使用 **peers=** 主机变量创建节点对节点连接。以下示例将 **control-plane-1.example.com** 连接到 **execution-node-1.example.com**，**execution-node-1.example.com** 连接到 **execution-node-2.example.com** ：

```
[automationcontroller]
control-plane-1.example.com peers=execution-node-1.example.com

[automationcontroller:vars]
node_type=control

[execution_nodes]
execution-node-1.example.com peers=execution-node-2.example.com
execution-node-2.example.com
```

其他资源

- 有关如何实施网格节点的更多示例，请参阅本指南中的自动化网格拓扑示例。

第 2 章 设置自动化网格

配置 Ansible Automation Platform 安装程序，为您的 Ansible 环境设置自动化网格。执行其他任务以自定义安装，如导入证书颁发机构(CA)证书。

2.1. 自动化网格安装

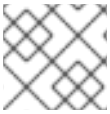
您可以使用 Ansible Automation Platform 安装程序设置自动化网格或升级到自动化网格。要为 Ansible Automation Platform 提供网格网络中节点、组和对等关系的详情，您可以在安装程序捆绑包的 **inventory** 文件中定义。

其它资源

- Red Hat Ansible Automation Platform 安装指南
- 自动化网格设计模式

2.2. 导入证书颁发机构 (CA) 证书

证书颁发机构 (CA) 在自动网格环境中验证并签署单个节点证书。您可以通过在 Red Hat Ansible Automation Platform 安装程序的 **inventory** 文件中指定证书的路径和私有 RSA 密钥文件的路径来提供自己的 CA。



注意

如果没有提供 CA，Ansible Automation Platform 安装程序会生成一个 CA。

流程

1. 打开 **inventory** 文件以进行编辑。
2. 添加 **mesh_ca_keyfile** 变量，并指定到私有 RSA 密钥 (**.key**) 的完整路径。
3. 添加 **mesh_ca_certfile** 变量并指定 CA 证书文件的完整路径 (**.crt**)。
4. 保存对清单文件的更改。

示例

```
[all:vars]
mesh_ca_keyfile=/tmp/<mesh_CA>.key
mesh_ca_certfile=/tmp/<mesh_CA>.crt
```

当添加到清单文件中的 CA 文件后，运行安装程序来应用 CA。这个过程将 CA 复制到每个控制并执行节点上的 **/etc/receptor/tls/ca/** 目录中。

其他资源

- Red Hat Ansible Automation Platform 系统要求

第 3 章 自动化网格设计模式

本节中的自动化网格拓扑提供了您可以在环境中设计网格部署的示例。示例包括一个简单的、hydrid 节点部署，到部署大量自动化控制器实例的复杂模式，采用多个执行和跃点节点。

先决条件

- 您检查了有关节点类型和关系的概念信息

注意

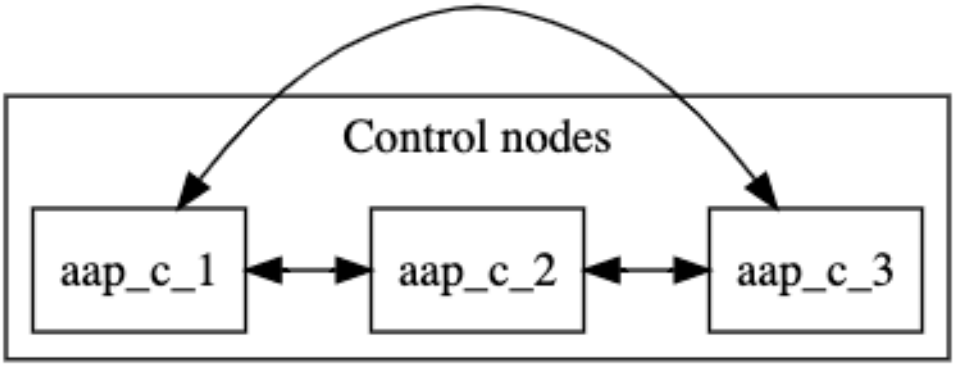
以下示例包括说明网格拓扑的镜像。镜像中的箭头表示对等方向。建立对等后，节点之间的连接允许双向通信。

3.1. 多个混合节点清单文件示例

这个示例清单文件部署由多个混合节点组成的 control plane。control plane 中的节点会自动相互连接。

```
[automationcontroller]
aap_c_1.example.com
aap_c_2.example.com
aap_c_3.example.com
```

下图显示了此网格网络的拓扑。



control plane 中节点的默认 **node_type** 是 **hybrid**。您可以在 **[automationcontroller group]** 中将单个节点的 **node_type** 明确设置为 **hybrid** ：

```
[automationcontroller]
aap_c_1.example.com node_type=hybrid
aap_c_2.example.com node_type=hybrid
aap_c_3.example.com node_type=hybrid
```

另外，您还可以在 **[automationcontroller]** 组中设置所有节点的 **node-type**。当您将新节点添加到 control plane 时，它们会自动设置为混合节点。

```
[automationcontroller]
aap_c_1.example.com
aap_c_2.example.com
aap_c_3.example.com
```

```
[automationcontroller:vars]
node_type=hybrid
```

如果您认为以后可能会将控制节点添加到 control plane 中，最好为混合节点定义一个单独的组，并为组设置 **node-type** ：

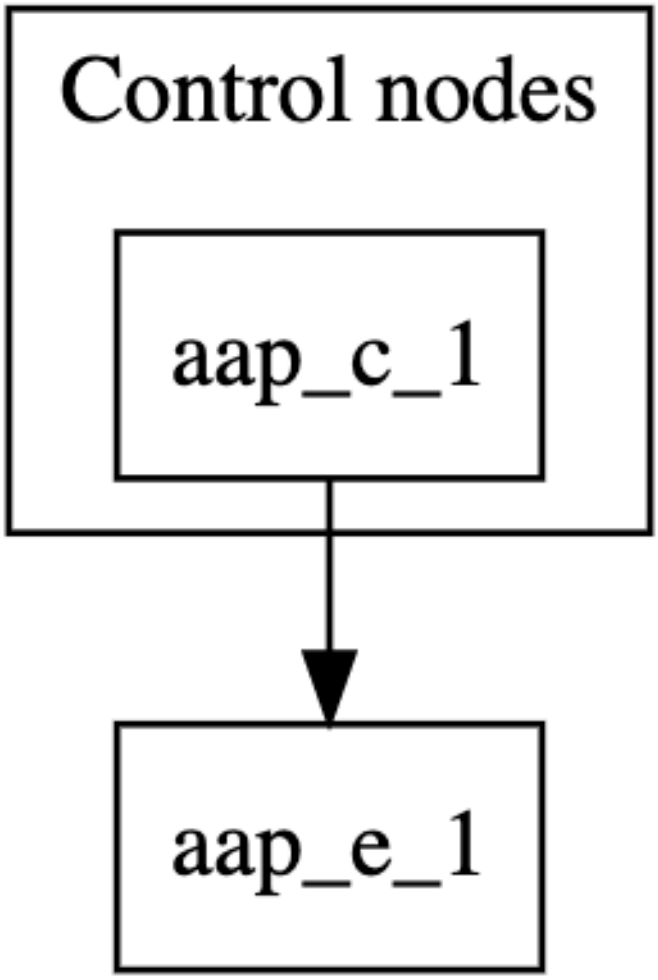
```
[automationcontroller]
aap_c_1.example.com
aap_c_2.example.com
aap_c_3.example.com

[hybrid_group]
aap_c_1.example.com
aap_c_2.example.com
aap_c_3.example.com

[hybrid_group:vars]
node_type=hybrid
```

3.2. 带有单个执行节点的单节点控制平面

这个示例清单文件部署单一节点 control plane，并与执行节点建立对等关系。

```
[automationcontroller]
aap_c_1.example.com

[automationcontroller:vars]
node_type=control
peers=execution_nodes

[execution_nodes]
aap_e_1.example.com
```

下图显示了此网格网络的拓扑。



[automationcontroller] 小节定义控制节点。如果您将新节点添加到 automationcontroller 组，它将与 **aap_c_1.example.com** 节点自动对等节点。

[automationcontroller:vars] 小节为 control plane 中的所有节点将节点类型设置为 **control**，并定义节点与执行节点对等的方式：

- 如果您向 **execution_nodes** 组添加新节点，则 control plane 节点会自动与其对等。
- 如果您向 **automationcontroller** 组添加新节点，节点类型被设置为 **control**。

[execution_nodes] 小节列出清单中的所有执行和跃点节点。默认节点类型是 **execution**。您可以为单个节点指定节点类型：

```
[execution_nodes]
aap_e_1.example.com node_type=execution
```

或者，您也可以在 **[execution_nodes]** 组中设置所有执行节点的 node_type。将新节点添加到组中时，它们会自动设置为执行节点。

```
[execution_nodes]
aap_e_1.example.com

[execution_nodes:vars]
node_type=execution
```

如果您计划将 hop 节点添加到清单中，最好为执行节点定义一个单独的组，并为组设置 **node_type** ：

```
[execution_nodes]
aap_e_1.example.com

[local_execution_group]
aap_e_1.example.com

[local_execution_group:vars]
node_type=execution
```

3.3. 最小弹性配置

这个示例清单文件部署由两个控制节点和两个执行节点组成的 control plane。control plane 中的所有节点都会自动相互连接。control plane 中的所有节点都与 **execution_nodes** 组中的所有节点相连接。此配置具有弹性，因为执行节点可从所有控制节点访问。

容量算法决定在启动作业时选择哪个控制节点。如需更多信息，请参阅*自动控制器用户指南*中的自动化控制器容量确定和作业影响。

以下清单文件定义了此配置。

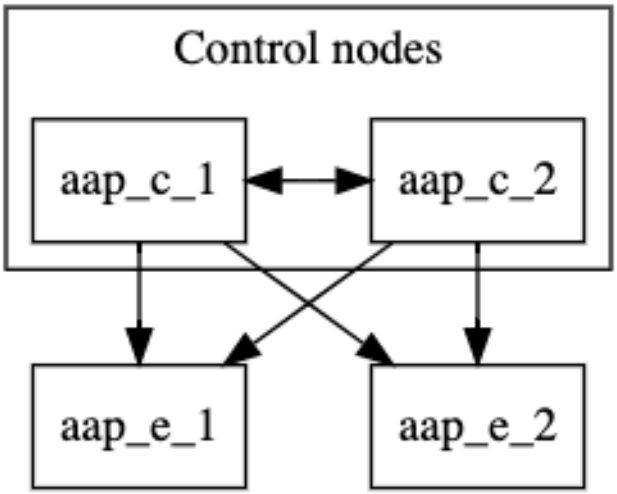
```
[automationcontroller]
aap_c_1.example.com
aap_c_2.example.com

[automationcontroller:vars]
node_type=control
peers=execution_nodes

[execution_nodes]
aap_e_1.example.com
aap_e_1.example.com
```

[automationcontroller] 小节定义控制节点。control plane 中的所有节点都相互对等。如果您将新节点添加到 **automationcontroller** 组，它将与原始节点自动对等。

[automationcontroller:vars] 小节为 control plane 中的所有节点将节点类型设置为 **control**，并定义节点与执行节点对等的方式：

- 如果您向 **execution_nodes** 组添加新节点，则 control plane 节点会自动与其对等。
- 如果您向 **automationcontroller** 组添加新节点，节点类型被设置为 **control**。

下图显示了此网格网络的拓扑。



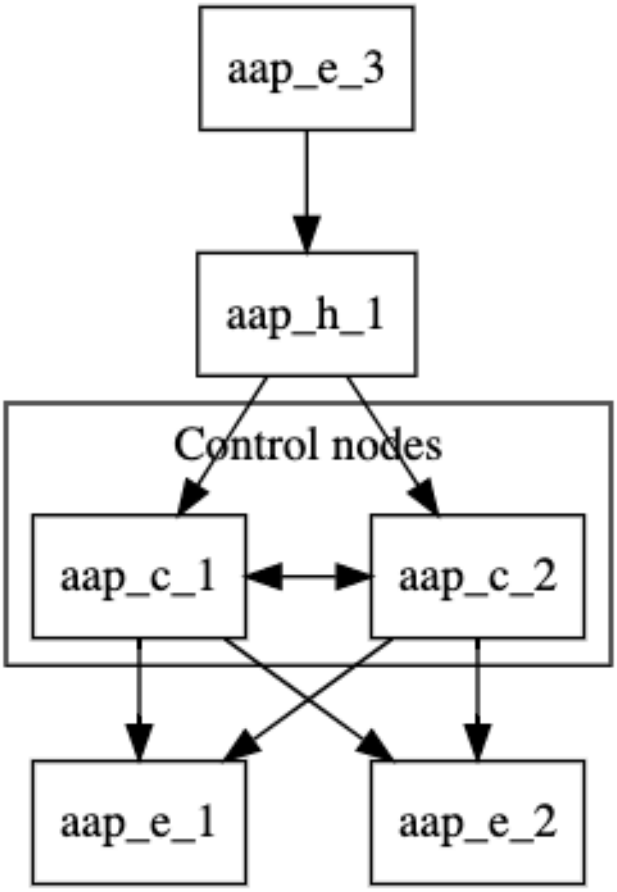
3.4. 隔离本地和远程执行配置

此配置会将跃点节点和远程执行节点添加到弹性配置中。可以从跃点节点访问远程执行节点。

如果要在远程位置设置执行节点，或者需要在 DMZ 网络中运行自动化，则可以使用此设置。

```
[automationcontroller]
aap_c_1.example.com
aap_c_2.example.com

[automationcontroller:vars]
node_type=control
peers=instance_group_local

[execution_nodes]
aap_e_1.example.com
aap_e_2.example.com
aap_h_1.example.com
aap_e_3.example.com

[instance_group_local]
aap_e_1.example.com
aap_e_2.example.com

[hop]
aap_h_1.example.com

[hop:vars]
peers=automationcontroller

[instance_group_remote]
aap_e_3.example.com

[instance_group_remote:vars]
peers=hop
```

下图显示了此网格网络的拓扑。



[automationcontroller:vars] 小节为 control plane 中所有节点设置节点类型，并定义控制节点与本地执行节点对等的方式：

- control plane 中的所有节点都会自动相互连接。
- control plane 中的所有节点都与所有本地执行节点相对等。

如果一组节点的名称以 **instance_group_** 开头，安装程序将它重新创建为实例组，并将它添加到 Ansible Automation Platform 用户界面中。

3.5. 多跃点执行节点

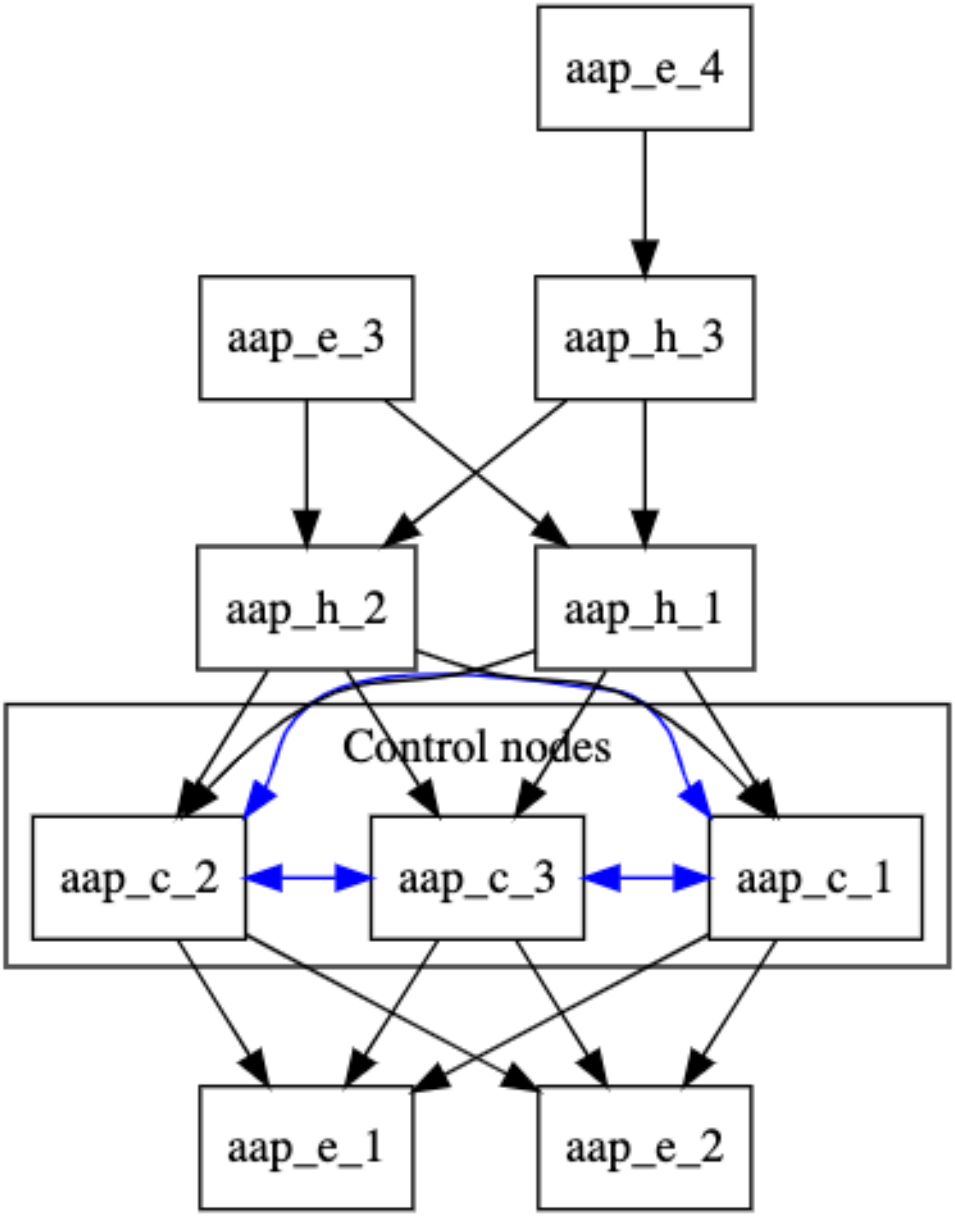
在此配置中，弹性的控制器节点与弹性本地执行节点相对等。弹性本地跃点节点与控制器节点的对等点。远程执行节点和远程跃点节点与本地跃点节点对等。

如果需要从远程网络在 DMZ 网络中运行自动化，您可以使用此设置。

```
[automationcontroller]
aap_c_1.example.com
aap_c_2.example.com
aap_c_3.example.com

[automationcontroller:vars]
node_type=control
peers=instance_group_local

[execution_nodes]
aap_e_1.example.com
aap_e_2.example.com
```

```
aap_e_3.example.com
aap_e_4.example.com
aap_h_1.example.com node_type=hop
aap_h_2.example.com node_type=hop
aap_h_3.example.com node_type=hop

[instance_group_local]
aap_e_1.example.com
aap_e_2.example.com

[instance_group_remote]
aap_e_3.example.com

[instance_group_remote:vars]
peers=local_hop

[instance_group_multi_hop_remote]
aap_e_4.example.com

[instance_group_multi_hop_remote:vars]
peers=remote_multi_hop

[local_hop]
aap_h_1.example.com
aap_h_2.example.com

[local_hop:vars]
peers=automationcontroller

[remote_multi_hop]
aap_h_3 peers=local_hop
```

下图显示了此网格网络的拓扑。



[automationcontroller:vars] 小节为 control plane 中所有节点设置节点类型，并定义控制节点与本地执行节点对等的方式：

- control plane 中的所有节点都会自动相互连接。
- control plane 中的所有节点都与所有本地执行节点相对等。

[local_hop:vars] 对 **[local_hop]** 组中的所有节点都带有所有控制节点的对等点。

如果一组节点的名称以 **instance_group_** 开头，安装程序将它重新创建为实例组，并将它添加到 Ansible Automation Platform 用户界面中。

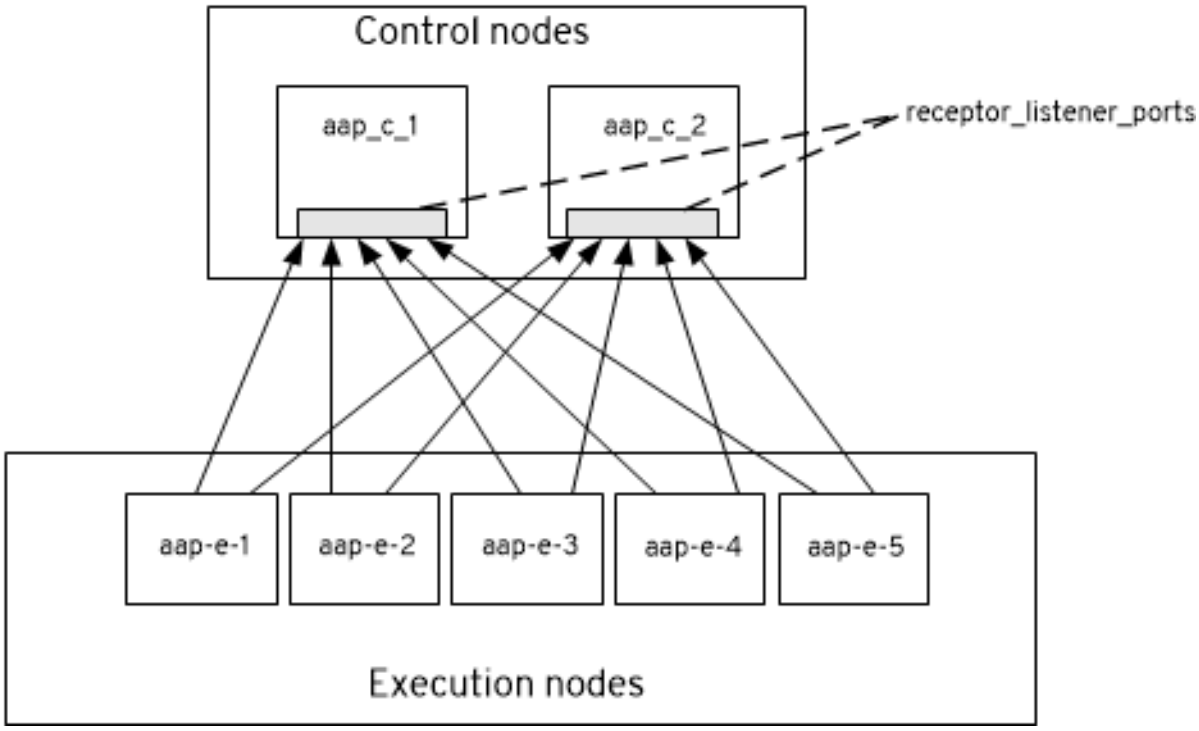
3.6. 到控制器节点的仅限出站的连接

这个示例清单文件部署由两个控制节点和多个执行节点组成的 control plane。只有到 'execution_nodes' 组中的控制器节点的出站连接才会与 controller plane 中的所有节点对等。

```
[automationcontroller]
controller-[1:2].example.com
```

```
[execution_nodes]
execution-[1:5].example.com

[execution_nodes:vars]
# connection is established *from* the execution nodes *to* the automationcontroller
peers=automationcontroller
```

下图显示了此网格网络的拓扑。



第 4 章 取消置备单个节点或组

您可以使用 Ansible Automation Platform 安装程序取消置备自动化网格节点和实例组。本节中的步骤描述了如何取消置备特定节点或整个组，以及每个流程的示例清单文件。

4.1. 使用安装程序取消置备独立节点

您可以使用 Ansible Automation Platform 安装程序从自动化网格中取消置备节点。编辑 **inventory** 文件，将节点标记为取消置备，然后运行安装程序。运行安装程序也会删除所有附加到节点的配置文件和日志。

注意

您可以取消置备任何清单的主机，但 **[automationcontroller]** 组中指定的第一个主机除外。

流程

- 将 **node_state=deprovision** 附加到您要取消置备的安装程序文件中的节点。

示例

这个示例清单文件从自动化网格配置中取消置备两个节点。

```
[automationcontroller]
126-addr.tatu.home ansible_host=192.168.111.126  node_type=control
121-addr.tatu.home ansible_host=192.168.111.121  node_type=hybrid  routable_hostname=121-addr.tatu.home
115-addr.tatu.home ansible_host=192.168.111.115  node_type=hybrid node_state=deprovision

[automationcontroller:vars]
peers=connected_nodes

[execution_nodes]
110-addr.tatu.home ansible_host=192.168.111.110 receptor_listener_port=8928
108-addr.tatu.home ansible_host=192.168.111.108 receptor_listener_port=29182
node_state=deprovision
100-addr.tatu.home ansible_host=192.168.111.100 peers=110-addr.tatu.home node_type=hop
```

4.1.1. 取消置备隔离的节点

您可以选择使用 **awx-manage** deprovisioning 程序手动删除任何隔离的节点。

警告

使用取消置备命令只删除没有迁移到执行节点的隔离节点。要从自动化网格架构中取消置备执行节点，请使用 安装程序方法。

流程

1. 关闭实例：

```
$ automation-controller-service stop
```

2. 从另一个实例运行取消置备命令，将 ***host_name*** 替换为清单文件中列出的节点的名称：

```
$ awx-manage deprovision_instance --hostname=<host_name>
```

4.2. 使用安装程序取消置备组

您可以使用 Ansible Automation Platform 安装程序从自动化网格中取消置备整个组。运行安装程序将删除附加到组中节点的所有配置文件和日志。

注意

您可以取消置备清单中的任何主机，但 **[automationcontroller]** 组中指定的第一个主机除外。

流程

- 将 **node_state=deprovision** 添加到与您要取消置备的组关联的 [group:vars]。

示例

```
[execution_nodes]
execution-node-1.example.com peers=execution-node-2.example.com
execution-node-2.example.com peers=execution-node-3.example.com
execution-node-3.example.com peers=execution-node-4.example.com
execution-node-4.example.com peers=execution-node-5.example.com
execution-node-5.example.com peers=execution-node-6.example.com
execution-node-6.example.com peers=execution-node-7.example.com
execution-node-7.example.com

[execution_nodes:vars]
node_state=deprovision
```

4.2.1. 取消置备隔离实例组

您可以选择使用 **awx-manage** deprovisioning 程序手动删除任何隔离实例组。

警告

使用取消置备命令只删除隔离的实例组。要从自动化网格架构中取消置备实例组，请使用安装程序方法。

流程

- 运行以下命令，将 ***<name>*** 替换为实例组的名称：

```
$ awx-manage unregister_queue --queuename=<name>
```