ZStack 技术白皮书精选

ZStack--通过 Ansible 实现全自动化

扫一扫二维码，获取更多技术干货吧

版权声明

摘要

大道至简·极速部署，ZStack 致力于产品化私有云和混合云。

ZStack 是新一代创新开源的云计算 IaaS 软件，由英特尔、微软、CloudStack 等世界上最早一批虚拟化工程师创建，拥有 KVM、Xen、Hyper-V 等成熟的技术背景。

ZStack 创新提出了云计算 4S 理念，即 Simple （简单）、Strong （健壮）、Smart （智能）、Scalable （弹性），通过全异步架构，无状态服务架构，无锁架构等核心技术，完美解决云计算执行效率低，系统不稳定，不能支撑高并发等问题，实现 HA 和轻量化管理。

ZStack 发起并维护着国内最大的自主开源 IaaS 社区——zstack.io，吸引了 6000 多名社区用户，对外公开的 API 超过 1000 个。基于这 1000 多个 API，用户可以自由组装出自己的私有云、混合云，甚至利用 ZStack 搭建公有云对外提供服务。

ZStack 拥有充足的知识产权储备，积极申报多项软著和专利，参与业内标准、白皮书的撰写，入选云计算行业方案目录，还通过了工信部云服务能力认证和信通院可信云认证。ZStack 面向企业用户提供基于 IaaS 的私有云和混合云，是业内唯一一家实现产品化，并领先业内首家推出同时打通数据面和控制面无缝混合云的云服务商。选择 ZStack，用户可以官网直接下载、1 台 PC 也可上云、30 分钟完成从裸机的安装部署。

目前已有 1000 多家企业用户选择了 ZStack 云平台。

ZSTACK--通过 ANSIBLE 实现全自动化

Agent 是一种常见的 IaaS 软件管理设备的方式；例如，ZStack 使用 Python agents 去管理 KVM 主机。因为海量的设备，安装和升级 agents 成为巨大的挑战，所以大多数 IaaS 软件把这个问题留给客户或分发商，从而导致解决方案变得脆弱，因为缺乏 IaaS 软件本身的支持。ZStack 从一开始就在考虑这个问题，先后尝试了 Puppet、Salt 和 Ansible，最后实现与 Ansible 无缝并对用户透明的集成。所有的 ZStack agents 通过 Ansible 自动部署、配置、升级；用户可能根本没有注意到他们的存在。

动机

IaaS 软件通常是一个包含很多小程序的组合软件。理想情况下，IaaS 软件可以被写成一个中央管理软件，可以通过设备的 SDK 和设备对话；但在现实中，设备要么没有提供 SDK，要么提供的 SDK 不完整，迫使 IaaS 软件必须部署一个叫 agent 的小程序去控制它们。虽然 ZStack 把所有服务打包在一个单一的进程中（详见“进程内的 Microservices 架构”），部分 agent 依旧需要部署到不同的设备上以控制它们。部署 agent 的过程中，不仅需要安装 agent 和相关依赖的软件，还需要配置目标设备，这并不简单，而且通常需要用户做大量的手动工作。当 IaaS 软件管理着大量的设备的时候，这个问题变得非常显著，甚至会限制数据中心规模。

问题

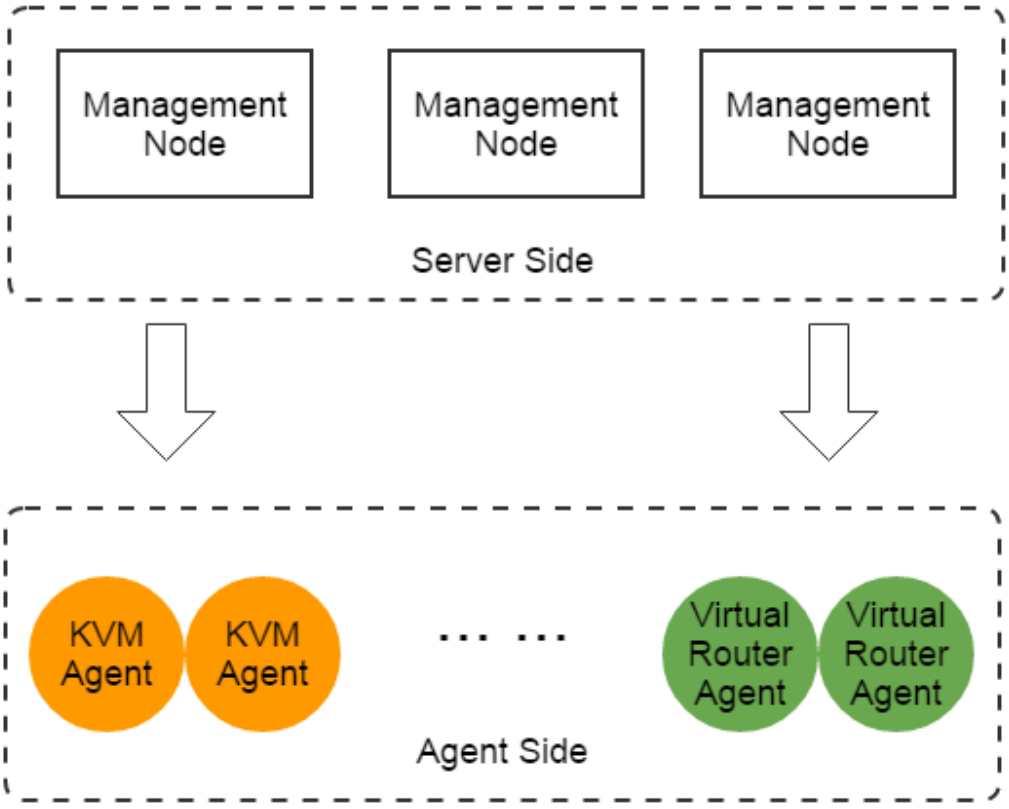
部署、升级 agent 以及配置目标设备的问题都属于配置管理问题，Puppet、Chef、Salt 和 Ansible 这类软件就是旨在解决这类问题。许多开发人员已经开始使用这些工具软件将 IaaS 软件包装成一个易于部署的方式；例如，为了安装 OpenStack，你可能会试图找到一些 puppet 模块而不是依据它提供的文档手动完成每一步操作。这些第三方包装能在一

定程度上缓解问题，但它们同时又是脆弱的，包装的软件发生任何变化都会破坏它们。另一方面，当用户想要配置软件包的某一部分的时候，他们可能不得不深入那些第三方包装去做一个即需的改变。最后，第三方包装无法处理封装的软件升级的状况，迫使用户重新面对他们试图隐藏的令人气馁的细节。

通过与配置管理软件 Ansible 无缝且透明的集成，ZStack 解决了这个问题。使用的方式是对用户隐藏细节并承担了管理 agent 的责任，最终展示给用户一个可以简单下载和运行（或升级）的软件，完全满足在数据中心自动化完成一切的目标，并帮助管理员克服安装、配置和升级云的恐惧。

和 Ansible 集成

ZStack 有一个典型的服务端-代理（server-agent）架构，服务器端包含所有驱动业务逻辑的编排服务，代理端执行来自于编排服务的命令，通过使用运行在不同的设备上的小的 Python agents（如 KVM agents、虚拟路由 agents）。



服务和 agents：*ZStack 对服务和 agents 有明确的定义。服务负责在云中完成一部分业务，例如存储服务。服务通常在 ZStack 管理节点运行的进程中，有自己的 API 和配置，与其他服务协同完成云上的业务。agent 是一个被服务命令的小附属程序，可以通过使用它来操作没有提供像样 SDK 的外部设备；例如，SFTP 备份存储 agent 在一台使用 SFTP 协议的 Linux 机器上创建了一个的备份存储。服务和代理的设计原则是****把所有复杂的业务逻辑放在服务中，使代理尽可能简单****。我们希望这个解释可以给你一个在 ZStack 中我们把什么称之为服务和代理的概念，因为其他 IaaS 软件可能有不同的想法，我们看到一些 IaaS 软件已经在 agent 代码中处理业务逻辑了。*

所有的 ZStack agents 包含三个文件：一个 Python 包（xxx.tar.gz）和 init.d 服务文件，和一个 Ansible YAML 的配置，在`$web_container_root/webapps/zstack/WEB-INF/classes/ansible`文件夹下它们自己目录中，所以一个服务可以通过 java classpath 找到它的 agent。Ansible

YAML 配置将所有的东西放在一起；它讲述了如何安装 agent，agent 的依赖，以及如何配置目标设备。引用 KVM 为例，其 Ansible YAML 配置中的一部分看起来像这样：

```
- name: install kvm related packages on RedHat based OS

  when: ansible_os_family == 'RedHat'

  yum: name=""

  with_items:

    - qemu-kvm

    - bridge-utils

    - wget

    - qemu-img

    - libvirt-python

    - libvirt

    - nfs-utils

    - vconfig

    - libvirt-client

    - net-tools



- name: install kvm related packages on Debian based OS

  when: ansible_os_family == 'Debian'

  apt: pkg=""
```

```
  with_items:
    - qemu-kvm
    - bridge-utils
    - wget
    - qemu-utils
    - python-libvirt
    - libvirt-bin
    - vlan
    - nfs-common


- name: disable firewalld in RHEL7 and CentOS7
  when: ansible_os_family == 'RedHat' and ansible_distribution_version >= '7'
  service: name=firewalld state=stopped enabled=no


- name: copy iptables initial rules in RedHat
  copy: src="/iptables" dest=/etc/sysconfig/iptables
  when: ansible_os_family == "RedHat" and is_init == 'true'


- name: restart iptables
  service: name=iptables state=restarted enabled=yes
```

```
      when: chroot_env == 'false' and ansible_os_family == 'RedHat' and is_init == 'true'



    - name: remove libvirt default bridge

      shell: "(ifconfig virbr0 &> /dev/null && virsh net-destroy default > /dev/null && virsh
net-undefine default > /dev/null) || true"
```

像上面展示的这样，这个 YAML 配置会关心对一个 KVM 主机的所有设置。你不需要担心 ZStack 会要求你手动安装很多依赖软件，并且不会收到任何由于缺乏依赖或者错误配置引起的奇怪的错误。让一切运行在你的 Linux 机器上是 ZStack 的责任，不管你的 Linux 操作系统是 Ubuntu 还是 CentOS，即使你只部署了一个最小的安装，ZStack 也知道如何让它们准备就绪。

在 java 代码中的服务可以在某个恰当的时机，使用 `AnsibleRunner` 去调用 Ansible 去部署或升级 agents。KVM 的 `AnsibleRunner` 看起来像：

```
    AnsibleRunner runner = new AnsibleRunner();

    runner.installChecker(checker);

    runner.setAgentPort(KVMGlobalProperty.AGENT_PORT);

    runner.setTargetIp(getSelf().getManagementIp());

    runner.setPlayBookName(KVMConstant.ANSIBLE_PLAYBOOK_NAME);

    runner.setUsername(getSelf().getUsername());

    runner.setPassword(getSelf().getPassword());

    if (info.isNewAdded()) {

        runner.putArgument("init", "true");
```

```
        runner.setFullDeploy(true);

    }

    runner.putArgument("pkg_kvmagent", agentPackageName);

    runner.putArgument("hostname",
String.format("%s.zstack.org",self.getManagementIp().replaceAll("\\.", "-")));

    runner.run(new Completion(trigger) {

        @Override

        public void success() {

            trigger.next();

        }



        @Override

        public void fail(ErrorCode errorCode) {

            trigger.fail(errorCode);

        }

    });
```

`AnsibleRunner`非常智能。它将跟踪每一个 agent 文件的 MD5 校验值，并在远程设备测试 agent 的端口连接性，保证 Ansible 只在需要的时候被调用。通常情况下，部署或升级的过程是对用户透明的，在服务定义的触发点被触发；例如，一个 KVM agent 将在添加一个新的 KVM 主机的时候自动被部署。然而，服务也提供叫做`Reconnect API`的 API，让用户用命令式的方式触发 agent 部署；例如，用户可以调用`APIReconnectHostMsg`让一个 KVM agent 重新部署，重新部署的原因可能是为 Linux 操作系统修复一个关键的安全漏洞，在他们对 KVM

YAML 配置做出了相应的改变之后。未来，ZStack 将提供一个框架，允许用户执行他们定制的 YAML 配置而不用修改 ZStack 的默认配置。

在软件升级过程中，用户在安装完一个新的 ZStack 版本并重启所有管理节点后，`AnsibleRunner` 将检测到 agents 的 MD5 校验和发生了变化，并自动在外部设备升级 agents。这个过程是对用户透明的且精心设计的；例如，主机服务如果它发现共有 10,000 台主机，将每次升级 1000 台 KVM 主机，以避免管理节点的资源被耗尽；当然，我们也为用户提供了全局配置去优化这个行为（例如每次升级 100 台主机）。

总结

在这篇文章中，我们演示了 ZStack 与 Ansible 的无缝、透明的集成。通过这种方式，agent 安装、配置和升级的过程是全自动的，让繁杂的细节远离用户，留给 ZStack 自身来处理。