Fleet Management of ztC Edge using REST APIs

Introduction

This Whitepaper is to detail Fleet Management of ztC Edge devices to enable remote monitoring and management of the ztC Edge servers in their environment using REST APIs.

Details include two possible approaches, either of which can be used for ztC Edge remote monitoring and management. Whitepaper also provides 2 possible reference architectures that can be further enhanced for addressing specific monitoring and management requirements.

Based on the specific factors, one of the approaches/architectures below can be used as a final approach.

Business Requirements

- Today, each ztC Edge device/cluster is enabled with a User Interface (UI) that can be used for extracting the information about the host/cluster related resource utilization, logs, hosted VMs, their creation, statistics, status etc.
- The sheer range of different VMs that the ztC Edge is qualified for makes it suitable for addressing many organizational requirements. Hence, the organizations can have multiples of ztC Edge devices in their portfolio.
- All of the ztC Edge devices can be managed as 2 node clusters only through UI, however when there are several ztC Edge devices deployed in a network, manual check by logging into each device/cluster's UI will be extremely inefficient and tedious. Fleet management is the way to go forward and provides ability to manage all of the devices/clusters in the network from a single console.
- ztC Edge as part of 3.0.0 introduced REST APIs to enable the ability to manage and monitor multiple clusters.

Key Concepts

For the success of a Fleet Management system, there is a need to discover and access the ztC devices. Push or Pull mechanism are the two ways to get the information related to the physical host and or VM changes/updates.

- Pull based systems
  - An agent (additional software) must be running on the device. Agents are configured for the required functionality like for e.g. after a specific time, agent will communicate with the server for required actions and pull the data from the server to perform the required functionality.
  - For monitoring and management, agent will collect all information in line with the previous server query and pass it back during its timed communication with the server.
  - Disadvantages: Requires an agent running on the system. Agent should interact with the system to get all the required data and present/communicate back in the required format
- Push based systems
  - Server communicates with the device and pushes the instructions for required action. There is no additional agent needed.
  - For monitoring and management, the server connects the device directly and runs the required commands/APIs to collect the details.
  - Disadvantages: Device IP addresses should be known by the server

This Whitepaper, refers to the Push based system where a central server will discover and then communicate with the ztC devices to get information required for monitoring and management of ztC devices

Solution Overview

Mission: Manifest a non-intrusive method/entity for monitoring and management of current ztC environment that includes host hardware and VMs with minimal to negligible customer impact.

Push based systems

The proposed reference architectures make use of push-based mechanism for getting the latest information on VM management and host hardware.

Push mechanism basically involves a system that enables a server to push or pull data to and from the client without the client prompting the server to do so. Push based systems involve long-established connection from client to server using Web Sockets, MQTT or SSE etc. It provides real time data through the usage of APIs. This ensures reduced latency and efficient resource utilization compared to the polling model that can lead to delays between data availability and its reception. However, Push based also pose challenges with respect to push APIs complexity, management of long-lived connections between client, server and scalability.

Push based systems – Advantages and ztC Edge specifics

- No agent required on remote device.
- Server in full control of all update processes and configurations.
- With ztC Edge 3.0.0, we introduced REST APIs. This has laid a foundation for enabling the customers to use REST APIs for automation of available ztC Edge functionalities. This automation can be invoked from a centralized server having access to all ztC Edge devices/clusters and hereafter is referred as Fleet Management.
- These REST APIs have been used to carve out a reference architecture for Fleet Management. This architecture can be used by the customer for developing the centralized monitoring and management automation of their fleet of ztC Edge devices/clusters i.e., Fleet Management.

Reference Architecture methods

Two methods considered for reference architecture are

- *Ansible* - Provides open-source automation that reduces complexity, runs everywhere and allows automation of virtually any task. Workflows with multiple REST API calls can be executed as part of a single ansible playbook.
- *Kubernetes* - An open-source container orchestration system for automating software deployment, managing and monitoring applications, scaling, rolling out changes to applications.

Note: The use case realization for ztC Edge Fleet management can be achieved using multiple approaches. The decision as to which approach is to be used can be determined based on multiple possibilities like customer specific requirements and or specific network considerations. This document refers the two approaches that were found most suitable for the given use case. They are discussed in the following sections.

ztC Edge Fleet Management – Reference Architecture using Ansible

ztC Fleet Management reference architecture approach using Ansible is demonstrated below. This approach makes use of Ansible playbooks for management and monitoring of ztC Edge devices from a central server.

Ansible

Ansible provides open-source IT engine that aids automation of application deployment, service orchestration, cloud provisioning, and other IT tools. Ansible is an automation tool popular for being simple, light weight, consistent, free and open source along with support for OpenSSH making it very secure.

Some of the Ansible features include:

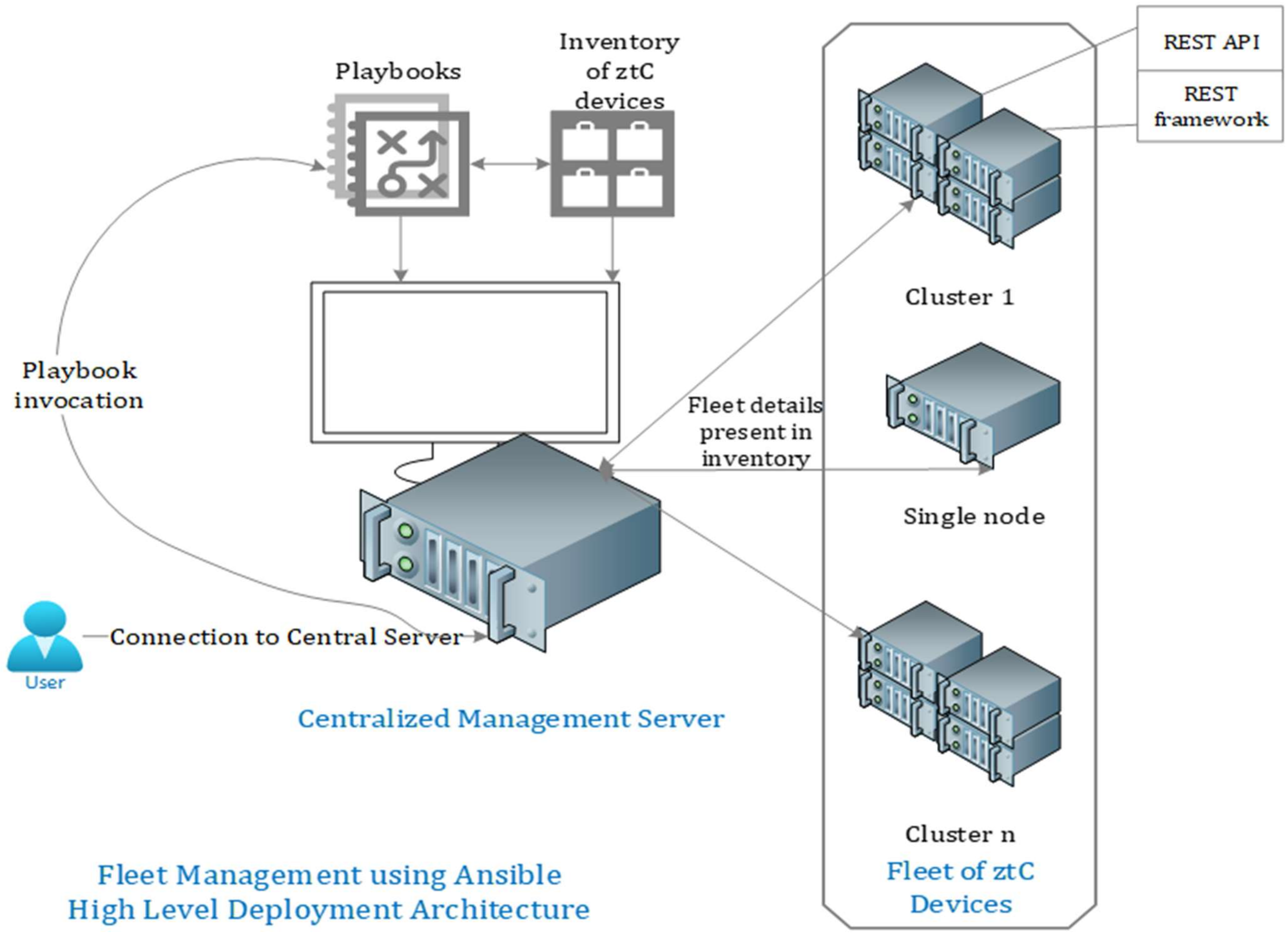
- An agent-less software that only needs SSH to connect the nodes to perform the required operations.
- It is built in Python and has a relatively low learning curve.
- It uses simple, human-readable scripts called playbooks to automate tasks.
- It can be used to deploy software on different servers simultaneously without human interaction.
- It enables us to create an automation of workflows implementation using more than one API call using a single playbook.

ztC Edge specifics

- Ansible is open-source suite of software tools that helps manage multiple machines using inventory from simple text files through an agentless architecture.
- Workflows can be scaled and deployed to multiple devices to aid in monitoring and management of various ztC Edge devices on the network. Workflows defining multiple REST calls can be defined as part of a single ansible playbook.

What & How:

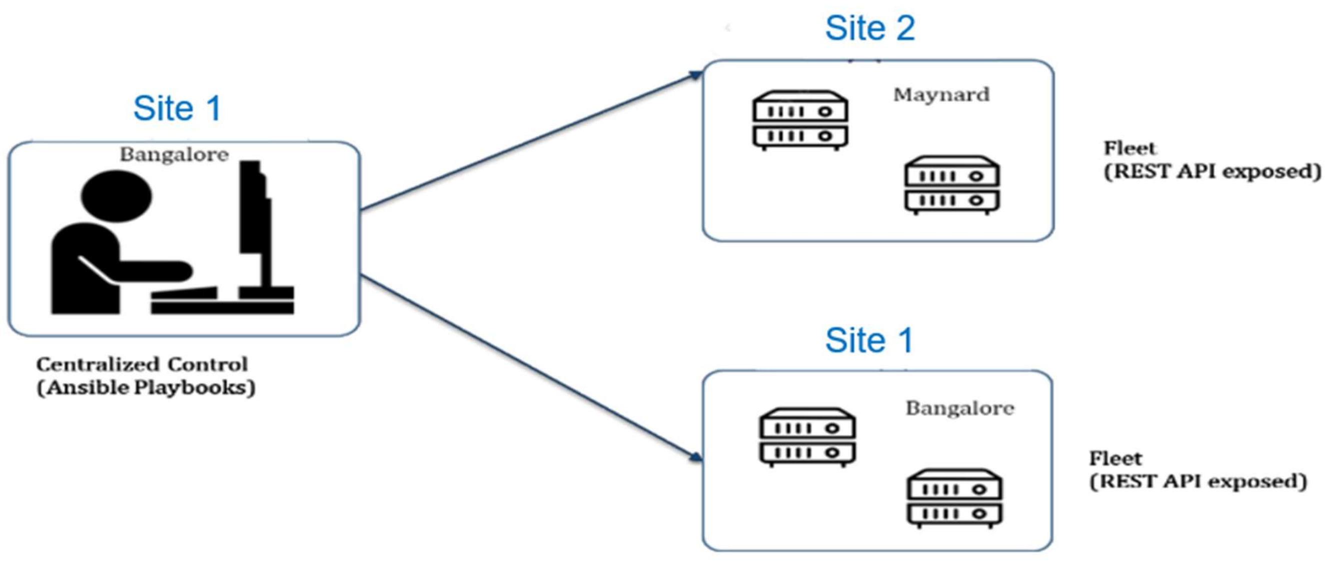
Below depicts high level deployment architecture of Fleet Management using Ansible. In this method, we have a central server which has the access to the inventory of available ztC Edge devices and has all required playbooks deployed which contain the calls to workflows that enable monitoring and management of these ztC Edge devices.


Playbooks
Inventory of ztC devices
Playbook invocation
Fleet details present in inventory
REST API
REST framework
Cluster 1
Single node
Cluster n
Fleet of ztC Devices
User
Connection to Central Server
Centralized Management Server
Fleet Management using Ansible
High Level Deployment Architecture

Following explains the different steps to be performed for using this approach

1. Central server should be deployed using Linux (Ubuntu22.04 was used in the lab).
2. All required packages should be installed on central server to enable successful execution of Ansible scripts.
3. Playbooks to be created for different workflows involving single or group of REST API calls to invoke VM management, monitoring of ztC Edge hardware.
4. Central server should have access to entire inventory of ztC Edge devices – IP addresses, required credentials.
5. Workflows involving multiple playbooks should be executed after generating the bearer token consuming user credentials and cluster IP addresses.
6. This process of invoking multiple playbooks can be scaled to any number of ztC Edge devices/clusters as per the requirements.

Below depicts demo setup done for PoC of Fleet Management using Ansible



ztC Edge Fleet Management – Reference Architecture using Kubernetes

Following section describes the ztC Fleet Management reference architecture approach using Kubernetes. This approach makes use of Kubernetes's master and pods concept for management and monitoring of ztC Edge devices.

Kubernetes

Kubernetes is an open-source cluster management software used for automation of software deployment, scaling and management.

Some of the Kubernetes features include:

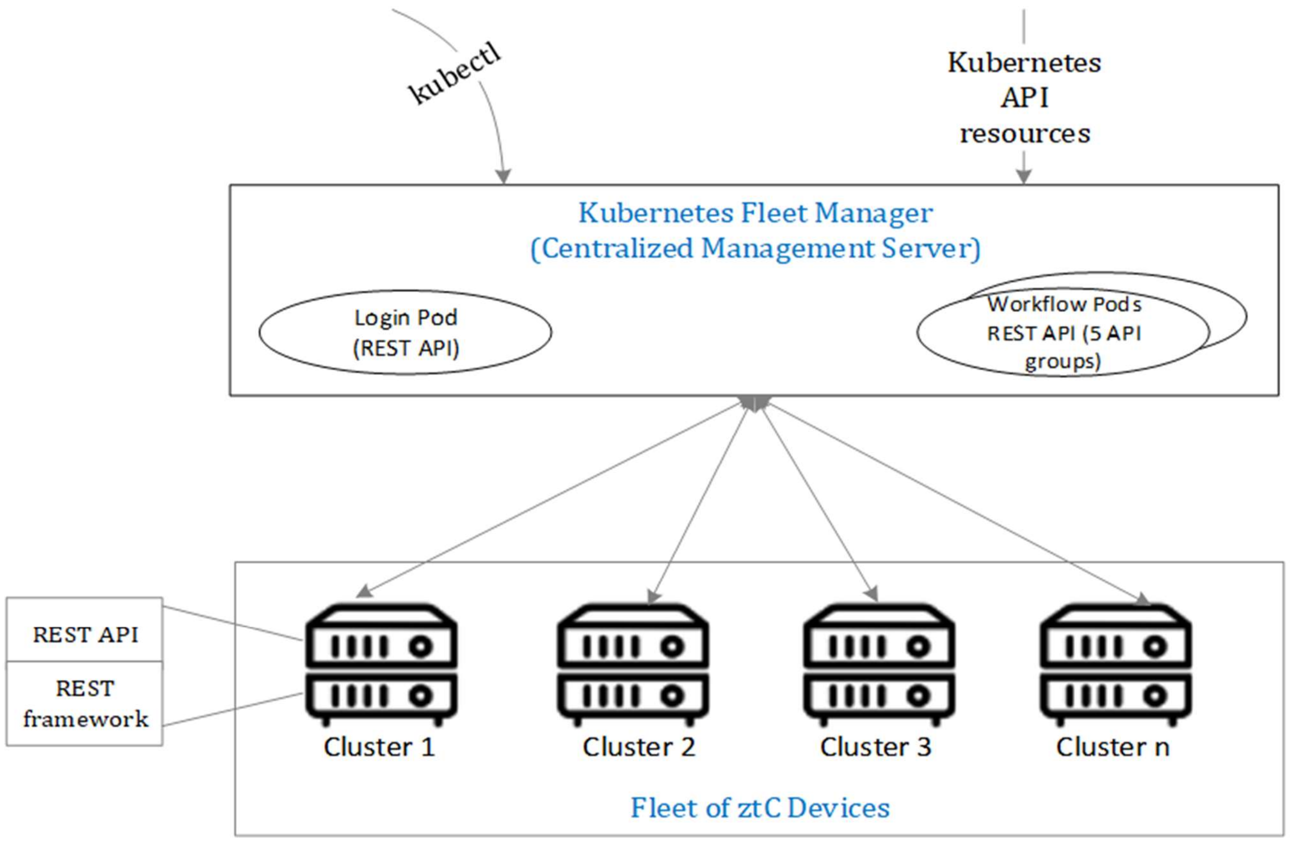
- Automates manual processes.
- Horizontal scaling.
- Health monitoring of nodes and containers.
- Provides security, networking, storage services including management of clusters, containers.
- Runs anywhere on-premises, public cloud infrastructure, hybrid allowing workloads to move as per user requirement.

ztC Edge specifics

- Kubernetes can be used to develop higher-level abstractions that allow rapid deployment of new apps, without understanding the internals of Kubernetes.
- Kubernetes aids in streamlining the workflow management. Workflows can be triggered using multiple pods (Login, specific workflow pod) which will aid in monitoring and management of various ztC Edge devices on the network.

What & How:

Below depicts high level architecture of Fleet Management using Kubernetes. In this method, we have a central management server which is the master. Pods get created/spawned/deleted based on the Workflow request



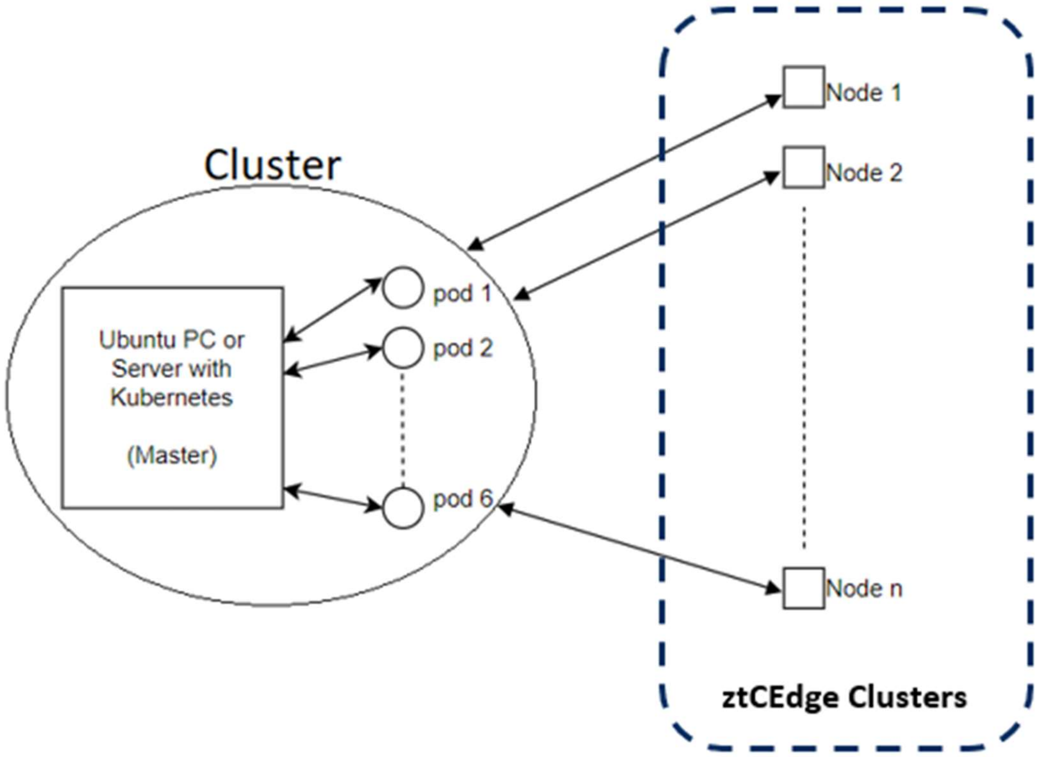
Fleet Management using Kubernetes
High Level Architecture

Following explains the different steps to be performed for using this approach

1. Central management server should be deployed using Linux (Ubuntu22.04 was used in the lab). This will be the Kubernetes Master node that will have pods for logging in to the ztC devices and for specific workflow that needs to be invoked.
2. Kubernetes cluster basically includes the Central server with the login pod and workflow pod.

3. All required packages (Kubernetes, docker) should be installed on central server to enable successful execution.
4. Workflows need to be created involving single or group of REST API calls to invoke VM management, monitoring of ztC Edge hardware so that they can be spawned as pods.
5. Central server should have access to entire inventory of ztC Edge devices – IP addresses, required credential

Note: After the token is generated and API pods are spawn, if any pod (login or API pod) gets terminated due to unforeseen event, the session is not lost and continues in the replica pod.



System and software used in lab

#	Ansible	Kubernetes
1	PC or server with Ubuntu 22.04.4 LTS	PC or server with Ubuntu 22.04.4 LTS
2	Ansible v2.11	Kubernetes v1.29
3	Python v3.11	Python v3.11
4		pwinput v1.0.1
5		Docker v23.03

Ansible vs Kubernetes

#	Ansible	Kubernetes
Purpose	It is a configuration management tool for IT infra-automation such as provisioning, configuration and deployment.	It is a container orchestration platform used for automating deployment, scaling and management of containerized applications.
Functionality	It can be used for managing infrastructure components, including servers, virtual machines, network devices etc.	It abstracts the infrastructure and provides a platform-agnostic way to deploy and manage the containers.
Tools and scripts	It includes YAML written playbooks that describe the state of the infrastructure, concentrating on tasks and configurations.	It uses declarative YAML manifests to define the state of applications, pods and other Kubernetes entities.
Deployment	It uses a push-based deployment model in which a control node pushes configurations and tasks to the managed nodes using a protocol like SSH.	User specifies the desired state of their applications and infrastructure using YAML manifests, Kubernetes ensures that desired state is achieved.
Learning curve	Simple to learn and manage the whole Ansible and its surroundings.	Management of Kubernetes cluster is tedious task and sometimes becomes overhead for quick turn around on issues and problems.

REST APIs in ztC Edge

ztC Edge 3.0.0 supports the Representational State Transfer (REST) application programming interface (API). This helps to collect system information, including physical machine (PM) and virtual machine (VM) properties, statistics, and alerts.

Provision of REST API and interactive REST API documentation is available through the OpenAPI Specification (Swagger).

For details on available APIs please refer ztC documentation REST API section using the following link zTC Edge REST APIs

Additional information

- This is only a *reference architecture*, which can be used as reference by customers and partners to implement Fleet Management. This is not *a product feature*.
- *Documentation and Demo videos* are published similar to the product documentation for public access at ztC Edge Fleet Management GitHub Repo
- Source code of *reference implementation* (Both Ansible and Kubernetes scripts) can be made available through *shared repository and controlled access*
- These are *not readily usable scripts*; users will have to develop their own implementation by using these as reference only.
- Users can decide to use *either of the approaches* or even use their own method to implement Fleet Management using REST APIs based on the need and expertise
- This is not available within the product nor published with product releases
- Contact Information: For any queries or help on further information and or access, please reach out to ztC Edge Fleet Management Support