vAssist - Voice based Multi-cloud Cloud Manager

Lakshmi Reddy
VMware
Email: reddyl@vmware.com

Sandeep K V
VMware
Email: sandeepkv@vmware.com

Amarnath Palavalli
VMware
Email: apalavalli@vmware.com

***Abstract*—*Multi-cloud Management* is increasingly gaining significance considering the rise of mega clouds like AWS, Azure, Google Cloud and many more. Self-service is the new mantra for cloud management. Hence, mega cloud vendors are focusing on end users to provide great user experience. However, the challenge still persists when it comes to learning multiple portals or complicated UI and workflows that comes with enterprise cloud management software. *vAssist* mobile app provides an extremely simple to use and consistent management interface for heterogeneous cloud based on popular personal assistant like *Google Now* on *Android* and same can be extended to other mobile OS platforms. *vAssist* app backend comes with common abstraction and predefined consistent set of commands that are interpreted and executed by a common cloud SDK capable of translating these commands into native cloud provider API calls. The app also comes with natural language processing capabilities to choose correct commands based on the voice input. Therefore, completely eliminates the learning curve and probably the documentation too, which aligns well will the motto of VMware MSBU to eliminate documentation and support for all management product offerings by offering great customer experience. We believe apps like *vAssist* can play a significant role in VMware's strategy of *Multi-Cloud and Multi-Device* by enabling the users to manage workloads across multiple mega clouds from anywhere.**

Keywords - Cloud Management, Android, Mobile App, Google Cloud

I. Introduction

IT users often find themselves servicing urgent and repetitive requests to create deployment environments with pre-specified configurations. Cloud automation and provisioning products help in creation of machines with pre-specified configurations easily. In VMware vRealize Automation product, this is achieved by allowing IT admins to model application blueprints based on desired configuration and then users deploying them from catalog of service blueprints providing self-service, across multiple cloud providers including private, hybrid and public clouds.

Public cloud providers are focusing more and more end users of IT services, offering simple to use self-service portal. However, the challenge is about vendor lock-in. Enterprise cloud management software vendors bridge this gap by providing common management interface for managing heterogeneous cloud. But the complicated workflows, significant learning curve and long implementation cycles make slower adoption.

Users of IT are increasingly embracing hand held devices like smart phones and tablets for all their service needs and enterprise IT is no exception. To ease IT users life, we propose *vAssist* app to extend the voice based personal assistant popular across the smart devices platform (like Cortana on Windows 10, Google Now on Android, Siri on iOS, etc.) to enable support for deployment of applications and services on heterogeneous cloud platform. Taking this one step further, the commands and notifications can be provided over social networking platforms like Twitter, WhatsApp, etc. Virtual personal assistant and social platform integration for cloud management provides great customer experience. This extends the VMwares vision of *any device and any cloud* to *from anywhere*.

The *vAssist* app comes with integration of *Natural Language Processing* capability and *Command Library* for predefined set of commands that can be consistently used across multiple crowd providers based on a common *Cloud SDK*. Therefore, *vAssist* can interpret the voice commands to suggest the available predefined commands for user to select in a easy and simple to use interface. Personal assistant and social apps are extremely popular among users and therefore no learning curve as well as may not require any documentation. This aligns with VMware MSBU's vision of no documentation for building *Multi-cloud Management* software that requires zero documentation and minimum support for user adoption.

One important aspect to note here is that, *vAssist* kind of applications can play crucial role in simplifying the life of end users of IT services by offering effective and simple to use interface for mundane and easy to use operations and dashboards, which forms the majority of usecases of any cloud management solution. However, they may not replace some of advanced tasks like creating complex application blueprints and workflows, that may still require rich web interface and bigger displays for better control and visualization.

II. Related Work

Spring Cloud[1] "provides tools for developers to quickly build some of the common patterns in distributed systems (e.g. configuration management, service discovery, circuit breakers, intelligent routing, micro-proxy, control bus, one-time tokens, global locks, leadership election, distributed sessions, cluster state). Coordination of distributed systems leads to boiler plate patterns, and using Spring Cloud developers can quickly stand up services and applications that implement those patterns. They will work well in any distributed environment, including the developer's own laptop, bare metal data centres, and managed platforms such as Cloud Foundry." [1]. We propose to build *vAssist Cloud SDK* on *Sprint Cloud* to provide

common API abstraction across cloud providers. It is also easy to extend the framework to add support for any cloud provider if the support does not exist.

Google *Cloud Console* app[2] enables users to manage their applications running on *Google Cloud Platform* from devices running *Android* OS. Users can monitor and troubleshoot their application using this app. However, the app is specific to *Google Cloud Platform* and can result into vendor lock-in. *vAssist* app offers the common set of capabilities to manage resources across several cloud providers.

Several mobile apps are available in the market to manage cloud storage services across various cloud storage providers. The popularity of these apps is rising due to easy of use, therefore adoption is rising rapidly. *vAssist* follows the similar direction for *Multi-cloud Management* capabilities on mobile apps including support for managing private clouds.

III. vAssist App

We have used *Google Cloud Platform* as our cloud provider platform to demonstrate *vAssist* app capability of managing *Multi-cloud*. The figure 1 provides the overview of *vAssist* app.

Fig. 1. vAssist App Overview

vAssist app comes with inbuilt intelligence to understand and interpret the voice command. Also, the app comes with *Natural Language Processing* capabilities and maps the input command to one of the standard predefined set of commands that work across all cloud providers.

nodeJS forms the platform for back end implementation of *vAssist* app running on cloud. We have chosen *Volley*[5] as protocol of communication between app and the back end. We looked into several alternatives like *okhttp*, *Retrofit* and few others as part of our initial study. All these protocols offer all the capabilities required for *vAssist*. However, *Volley* offers some differentiating features like queuing of requests, caching of resources and slightly better performance while handling multiple requests.

vAssist CloudSDK running on *nodeJS* server intercepts the requests from app and invokes the appropriate commands on the cloud provider (in this case *Google Cloud Platform*) and returns the response back to app. The communication between the *CloudSDK* and cloud provider is asynchronous, where in response is managed by callback handlers. Whereas, the communication between app and its back end is synchronous.

A. Architecture

Figure 2 gives insights into overall *vAssist* components and architecture.

vAssistant takes voice input from user and converts it to text with help of personal assistant based on the mobile OS platform (Google Now or Siri or Cortana), which forms the input for *Command Resolver*. The *Command Resolver* then looks up for supported commands from the *Command Library*, which comes with standard predefined set of commands supported across all cloud providers managed by *vAssist*. If no matching command found then it uses *NLP/Dictionary* module to look up for synonyms or equivalent text to see if there is a matching command available. If more than one match found with *NLP/Dictionary* lookup, then the list of possible options will be provided to user for selection. Once user makes the choice, it is forwarded to *Cloud SDK*. *Cloud SDK* provides abstraction to various cloud providers like vCloud Air, AWS, Google Cloud Platform, Microsoft Azure, etc. Also, it is quite possible to have VMware vRealize Automation (vRA) as cloud provider in case users have deployed for managing the cloud. This module invokes the API of cloud provider to execute the command and presents the results back to user. Flowchart shown in Figure 3 depicts the command execution flow.

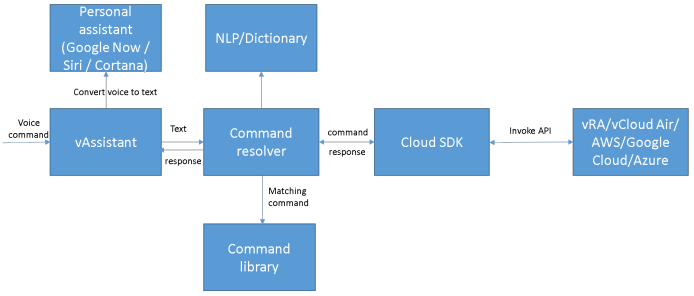

Fig. 2. Architecture - Voice based Cloud Management

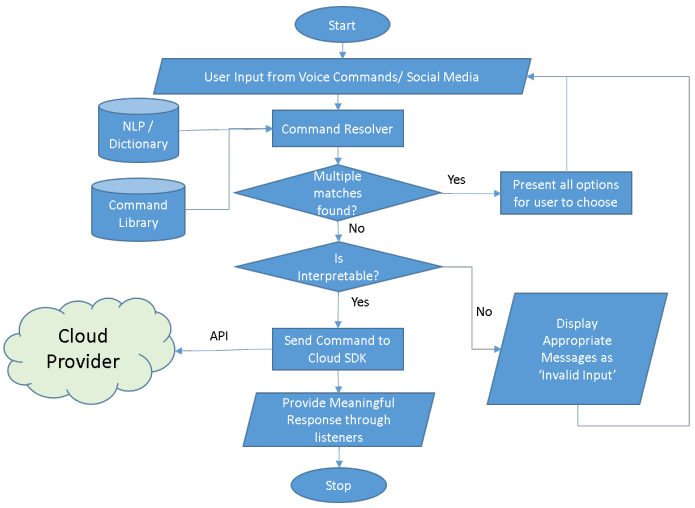

Fig. 3. vAssist Command Execution Flow

B. Usecases

User then gives a voice command for the operation to be performed. For example, *Create a VM*, then *vAssistant* takes the conversation forward until the requested action or operation completed. Currently *vAssist* supports the following operations.

- Create VM(s) - Capable of creating a VM on all cloud providers supported by *Cloud SDK*. This operation can be performed with default settings for a quick deployment of new VM or user can choose to customize. Also, it is possible to deploy multiple VMs with same configuration.
- List VMs - This Operation gives the list of Virtual Machines with basic info (status, os, machine type) about every vm.
- Start/Stop VM(s) - This operation lets the user to start a vm, stop a vm, start all the stopped vms, stop all the running vms.
- VM Info - This Operation gives all the information about a particular virtual machine, like status, machine type, zone, IP address,External IP etc.
- Delete VM(s) - This operation lets the user to delete a vm, delete group of stopped vms, or group of vms with some property, or delete all the vms.
- Health of VM(s) - This operation gives the information about the heath status of vm or group of vms.
- Summary for VM(s) - Shows the VM usage stats.
- List Templates - This operation gives info about different types of templates available to create a vm.
- List Machine Types - This opertion gives info about the different machine types (Num of cpus and vcpus, RAM Size, num of persistent disks) available to create vm.

Additionally, as part of *vRealize Automation* integration, the capabilities supported will include the following commands as illustration.

- Blueprint Catalog - Shows the catalog.
- Deploy Blueprint - Deploy a *Blueprint*.
- Create Blueprint - Create a *Blueprint*.

We have plans to support more advanced usecases that are discussed in the *Future Work* section.

C. Usecase 1 - Create VM

vAssist app provides an interface to user to take the voice commands as shown in Figure 4, user can activate this with a touch on *vAssistant* button.

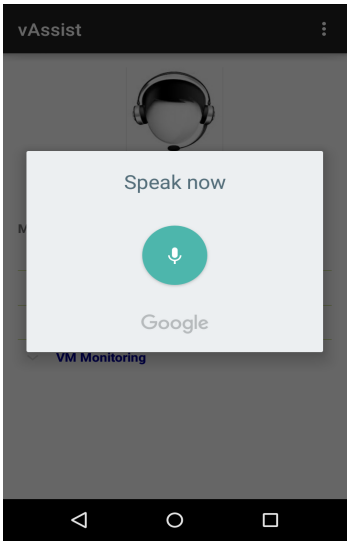

Fig. 4. Voice Command Input

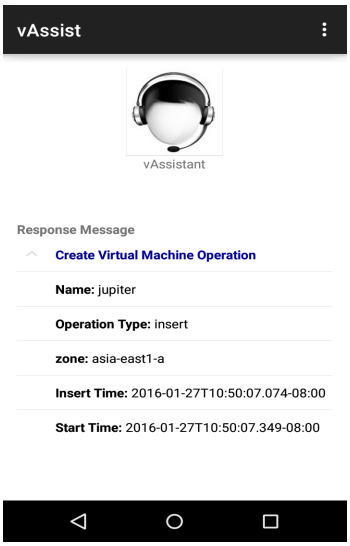

Fig. 5. Create VM Response

User presses, vAssistant Button,
vAssistant: Hello, What can I do for you?
User: Create a vm
vAssistant: I can create a Virtual Machine for you. Can you please tell me the name of the VM to be created?
User: jupiter
vAssistant: Ok, You chose your VM name as 'jupiter'. Do you want to create with Default Settings? or Custom Settings?
User: Can you tell me more about those?
vAssistant: Ok, I am happy to tell you about the Default and Custom Settings. In Default Settings the Operating System will be debian and the Machine Type will be n1-standard-1. That will have Standard 1 CPU machine type with 1 virtual CPU and 3.75 GB of memory. In Custom Settings, you can configure the Operating System and Machine Type.
So Now tell me what do you want? Default Settings? or Custom Settings?"
User: I am Ok with the default Settings.
vAssitant: Ok Cool.. I will create a virtual Machine with name 'jupiter' with default Settings.
Refer to Figure 5.

D. Usecase 2 - List VMs

User: Get me the list of VMs in my cloud
vAssistant: Sure. Let me get that information for you.
After Cloud SDK Response
vAssistant: Here is the list of VMs in your cloud.
Refer to Figure 6.

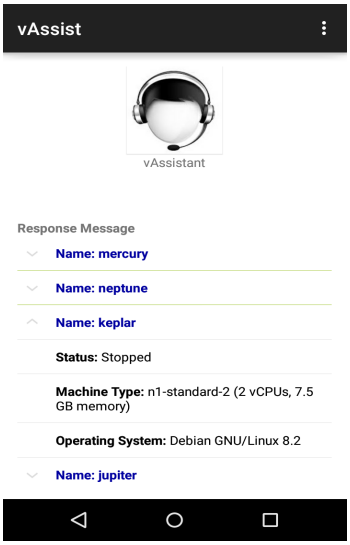

Fig. 6. VM List

E. Usecase 3 - VM Info

User: I want information about a vm
vAssistant: Sure. Please tell me the name of the vm you wanted the information
User: jupiter
vAssistant: Sure. Let me get the information about the vm 'jupiter' for you.
(After Cloud SDK Response)
vAssistant: Here is information about the vm 'jupiter'
Refer to Figure 7.

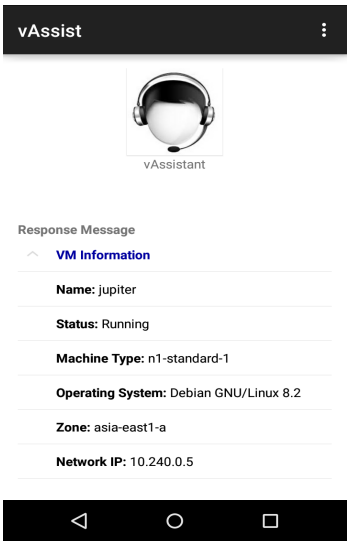

Fig. 7. VM Info

IV. Future Work

A. Social Networking Apps Integration

We would like to extend the benefits of *vAssist* to any consumer of infrastructure by integrating with popular social networking apps.

1) Twitter: Figure 8 shows the high level overview of components to extend capabilities of *vAssist* to integrate with *Twitter*.

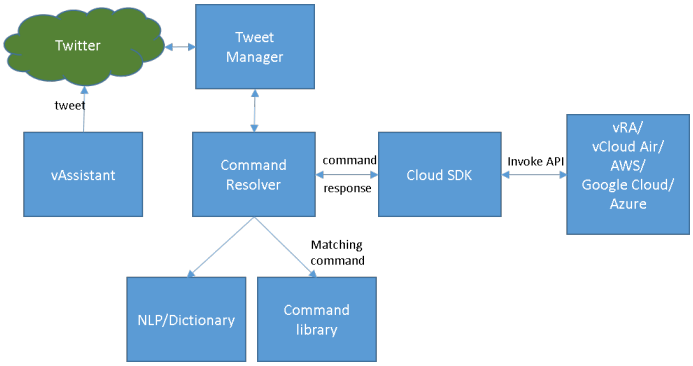

Fig. 8. Twitter Integration

Tweet Manager monitors Twitter account associated with the infrastructure to be managed. Users will send command Tweets to this account using *vAssistant*. On receiving the command *Tweet Manager* uses *Command Resolver* to interpret the command and invokes cloud provider APIs through *CloudSDK* to full fill user request. Finally, *Tweet Manager* posts command response as reply on the command Tweet. An example command Tweet to create two instances of SUSE Linux 32-bit machines can be given as
Create machine @Template:SUSE-Linux-32bit @NumInstances:2
Authorization - The Listener accepts commands only from users being followed by this twitter account. This ensures that the admin of twitter account can control which users can perform infrastructure operations.
Message Size - Twitter has limit on the message size and it is quite possible that a command tweet can exceed the size. We propose a technique where in a command can span across multiple tweets and Listener has the capability to wait for the batch of tweets in case of incomplete command. For example semicolon(;) can be used to terminate the command.

2) WhatsApp: In same lines of integration with *Twitter*, we propose integration of *vAssist* using a dedicated *WhatsApp* group. The group admin can add the users to this group enabling them to send commands as messages on this group.

3) Calendar: It is quite common for cloud admins to perform repetitive tasks at regular periods like add/remove/modify resources allocated to various applications, deploy/undeploy workloads, scale out and many more. An example can be, add more nodes to *accounting* application running on cloud during financial year closing period to handle additional load and scale down after this period. Integrating *vAssist* with *Calendar*

app, users can create the calendar events with embedded commands. *vAssit* can subscribe to *Calendar* notification and execute the commands using *Cloud SDK*. For example, user schedules a calendar event to create a machine with name 'VM1' at 9 pm today and also schedules another calendar event to release the resources allocated in 24 hours.

4) Cloud SDK: We are planning to build *Cloud SDK* based on *SpringCloud* [1] for our cloud manager app. *SpringCloud* provides a very nice abstraction to invoke APIs across popular mega cloud providers and comes with good community support. Today the support for *vCloud Air*, *vSphere* and *vRealize Automation* is missing from *SpringCloud*. However, it is in the interest of VMware to implement *SpringCloud* support for all our products, which will help to penetrate into large dev community in adopting VMware's cloud APIs.

5) Natural Language Processing: To interpret the voice commands in meaningful way we propose to integrate *vAssist* with popular *Natural Language Processing* framework.

V. Conclusion

It is important for VMware to position its products for end consumers of IT and penetrate into developer community to stay relevant in the changing times as IT organizations are increasingly becoming brokers of IT services, therefore self-service is the mantra for the future. We believe applications like *vAssist* can bring users of IT closer to VMware products and APIs, there by offering world class products and services that require no additional learning curve and eliminates the documentation altogether. On the engineering side, the company can benefit largely by avoiding development of product UIs in silos across engineering teams and focus on building consistent and easy to use APIs. Applications like *vAssist* can be the consistent and standard UI to all products and services from anywhere, thereby offering great customer experience. This can even open up rich ecosystem of applications on VMware APIs by partners, customers and developer community.

vAssist brings the ability of managing multiple mega clouds from a simple to use app on an end user device can boost VMware's vision of building products for *Multi-Cloud and Multi-Device*. Also, integration with *Spring Cloud* can increase the span into developer community to effectively leverage VMware products to build their solutions and services.

VI. vAssist App Demo Video Link

A quick video of *vAssist* app demonstrating the usecases described in this paper is available on here. vAssist Demo, alternatively copy paste this to browser (https://www.dropbox.com/s/cktp8z6jvwthw5f/PoC.mp4?dl=0).

References

[1] Project Spring Cloud
[2] Google Cloud Console
[3] Android
[4] nodejs
[5] Android Volley