Invention Disclosure No.: CN8-2016-0258

API Connect for Software on Cloud

Contact: Tao N Zhang/China/IBM

Inventors

- Tao N Zhang/China/IBM
- Wu Mi WM Zhong/China/IBM
- Xin Peng Liu/China/IBM
- Yuan Yao Deng/China/IBM
- Jian Wu Dai/China/IBM
- Qian DV Du/China/IBM
- Xi XQ Qiao/China/IBM

Problem statement and background

What is the problem solved by your invention?

More and more software applications want to be published as service and be delivered to the end user on cloud. Most of software sizes are large and need be provisioned by using virtual machine or container.

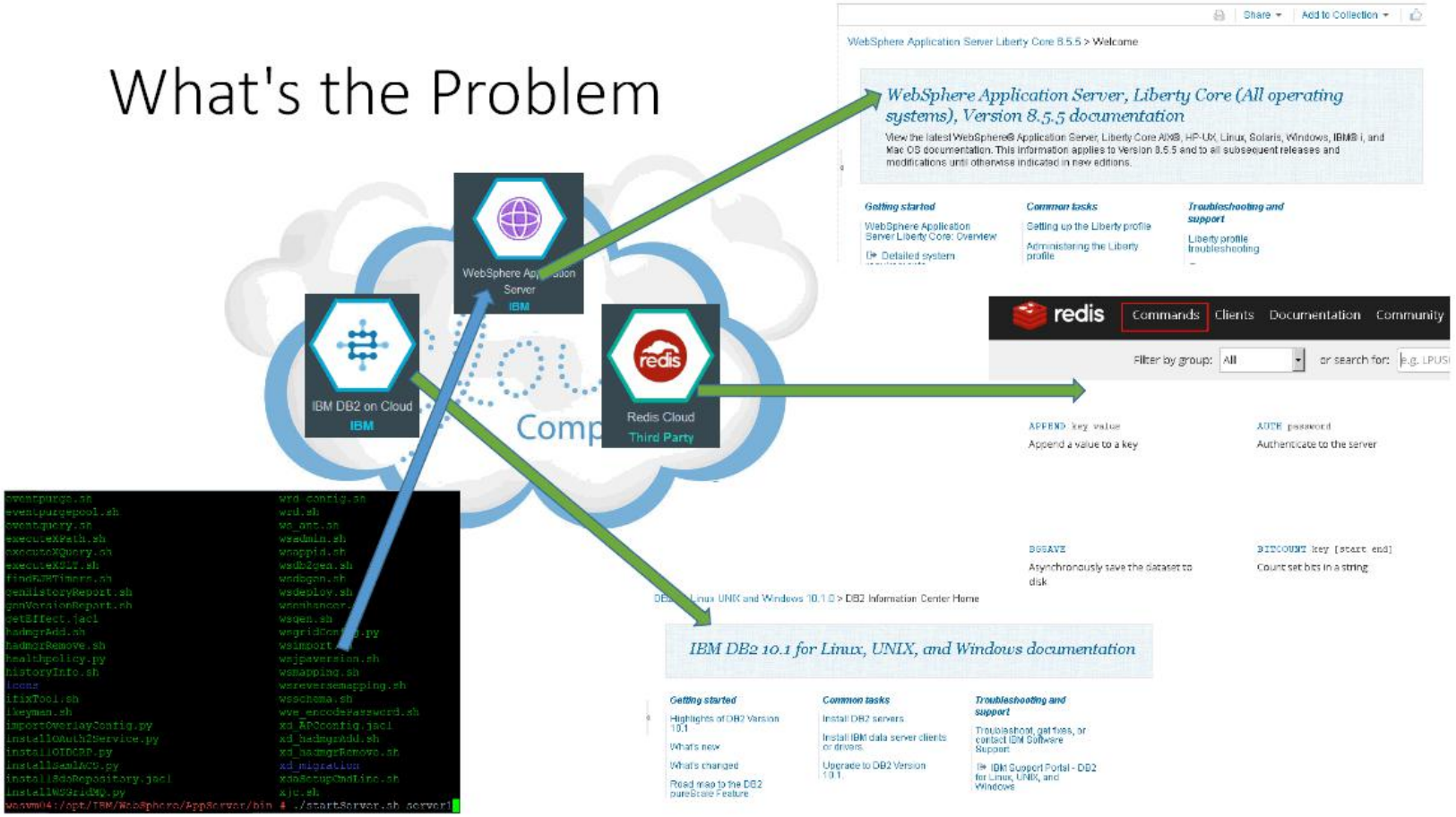
However in real cloud environment, customers always face a dilemma about how to use the software service easily:

1. For new user, he doesn't know where to run/how to use software on cloud immediately.

2. Until now, cannot only expose approved commands to end-user when using VM or container, user has to set complex security control to limit service/command usage.

3. User has to login Cloud VM or even use VPN to operate software, such as start, stop instance or collect log.

Problem statement and background



In summary, current service solution is not perfect in real cloud environment, which makes service use complex. We need a new methodology to guarantee the software service can be used more efficient and easier on the cloud environment.

Core idea/Claims

By scanning a service image, a list of all executable shell command of this service can be collected. Those commands exists for service operations. In this disclosure, we generate REST API for each of detected command and expose them to the end user, so that service on cloud will no longer require to expose their operation console directly so that the service is more secured since we only expose the necessary api to them.

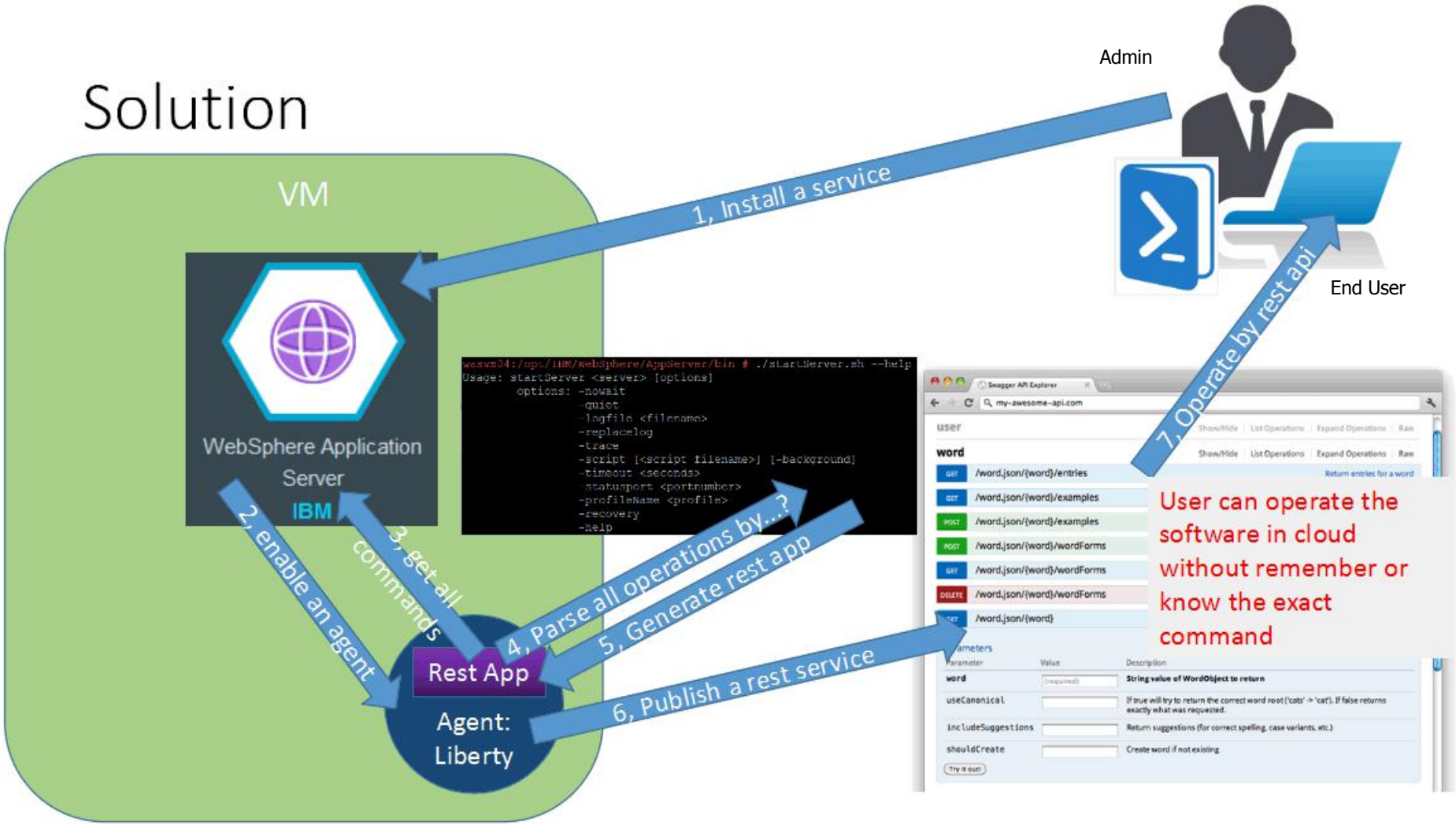
Adding to that, this disclosure can be implemented as a plugin (for example, a feature in Liberty) so that it can be plugged during the service provisioning.

what claimed:

A system and method to detect and expose executable command as RESTful services

- A system and method to discover interface-identifiable executable command from a (virtual) machine
- A system and method to wrap, bridge and expose interface-identifiable executable command to RESTful services.

Detailed Description



Detailed Description

Step 1, 2: Install service and install agent image in provisioning stage

[Provide a new container agent]

A checkbox is added in cloud service installation page, if user enable this "API Connect for service", cloud framework will download agent image and install the agent service by using Maven.

Step 3: Scan the software service all folders and get all commands list:

- By default, the software service has been installed during provisioning stage, so this software service environment is a controlled environment, This idea uses Maven to check if the file is executable command by using the condition:
- If the file properties value=755 or the file type is registered in OS.
- Then get the executable command list.

Detailed Description

Step 4: Generate JAX-RS resource code for each command:

....
startManager.sh
startNode.sh
startServer.sh
stopMWServer.sh
stopManager.sh
stopNode.sh
stopServer.sh
syncNode.sh
....
versionInfo.sh
....

```
public class StartServerResource {
    @GET
    @Path("startServer")
    public String startServer(@QueryParam("parameters") String parameters) {
        String result = "";
        Runtime run = Runtime.getRuntime();
        try {
            Process p = run.exec("<tWAS Folder>/bin/startServer.sh " + parameters);
            BufferedInputStream in = new BufferedInputStream(p.getInputStream());
            BufferedReader inBr = new BufferedReader(new InputStreamReader(in));
            String lineStr;
            while ((lineStr = inBr.readLine()) != null)
                result += lineStr;
            if (p.waitFor() != 0) {
                if (p.exitValue() == 1)
                    result = "Execute startServer error";
            }
            inBr.close();
            in.close();
        } catch (Exception e) {
            result = e.getMessage();
        }
        return result;
    }
}
```

4.1 This idea uses Maven to generate JAX-RS resource code by using a defined template.

4.2 Get option list from command help

Most executable commands support a “--help” option that displays a description of the command's supported syntax and options.

For the other commands don't support the “--help” option, but try to run it anyway. Often it results in an error message that will reveal similar usage information.

Detailed Description

Execute command with --help/-h to get all command help output message. Help message consists of 2 parts:

Usage pattern, e.g.:

Usage: app_program <arguments> [-hso FILE] [--quiet | --verbose]

Usage pattern is a substring of doc that starts with usage: (case insensitive) and ends with a visibly empty line.

Each pattern can consist of the following elements: <arguments>, [ ], |, ...

Option descriptions, e.g.:

```
-h --help     display this help message
-s --sorted   sorted output
-o FILE       specify output file [default: ./test.txt]
--quiet       suppress compiler output
--verbose       be extra verbose
```

Option descriptions consist of a list of options that you put below your usage patterns.

It is necessary to list option descriptions in order to specify:

* Synonymous short and long options: starts with - or -- (not counting spaces)
* If an option has an argument: put a word describing that argument after space (or equals "=" sign)
* If option's argument has a default value: [default: <my-default-value>]
* Use two spaces to separate options with their informal description

Their format is described below; other text is ignored.

Detailed Description

4.3 Get the list of options and replace the default parameter in template code and also add command usage description in rest list description:

```
@GET
@Path("startServer")
public String startServer(@PathParam("server") String server, @QueryParam("h") String h, @QueryParam("s") String s
    String result = "";
    Runtime run = Runtime.getRuntime();
    try {
        Process p = run.exec("<tWAS Folder>/bin/startServer.sh " + server + h + s + o + quite);
        BufferedInputStream in = new BufferedInputStream(p.getInputStream());
        BufferedReader inBr = new BufferedReader(new InputStreamReader(in));
        String lineStr;
```

Then you can access the command by using the generated rest resource:


```
ADMU0116I: Tool information is being logged in file
/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/logs/server1/startServer.log
ADMU0128I: Starting tool with the AppSrv01 profile
ADMU3100I: Reading configuration for server: server1
ADMU3200I: Server launched. Waiting for initialization status.
ADMU3000I: Server server1 open for e-business; process id is 11716
```

Detailed Description

4.4 Obtain service admin information when provisioning and add authentication check for the rest service invoking.

Add authentication configuration in both rest application and agent configuration by obtaining the cloud user's authorization which mapped to administrator account of the corresponding operation automatically.

Can also provide a configuration file to list the commands which want to be excluded.

4.5 Package all rest resources and generate application war file.

Step 5: Export the generated app and deploy to agent

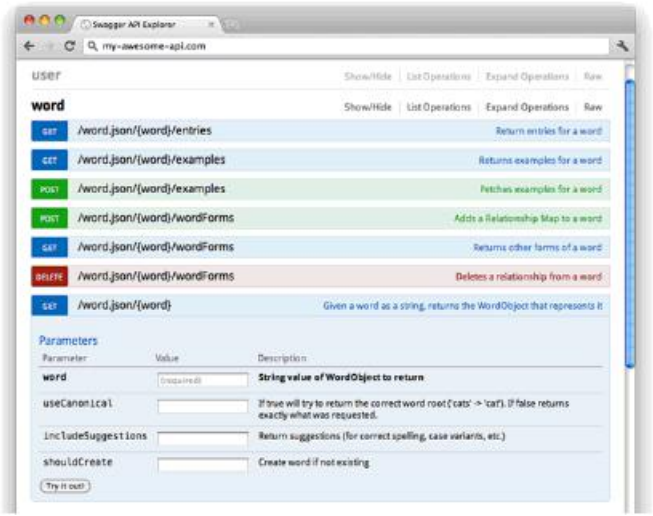
Distribute this application war file to the agent application folder.

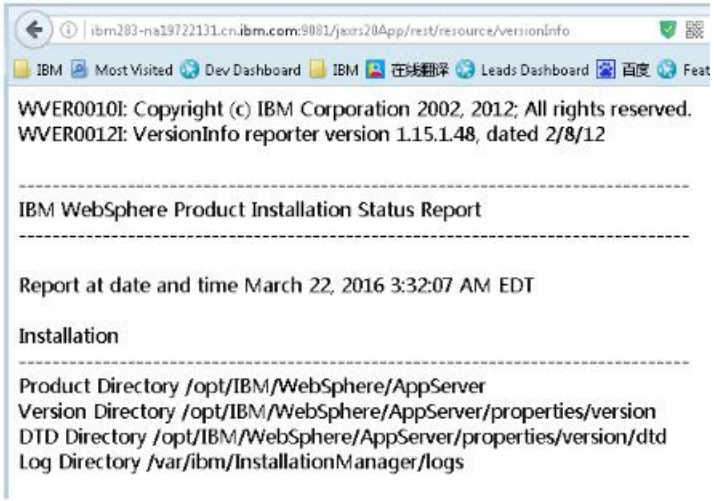
Agent can discover rest resource without any coding or operation, such as Swagger Maven plugin or enable Liberty API Discovery feature, then agent can discover all rest resources and expose them on a special API URL.

Detailed Description

Step 6: Auto start agent to provide API connect service

After finish software service provisioning and API connect agent setup, start agent and rest application then publish the API rest service to cloud by using Maven plugin.
Then user can view the rest API list and use this API Connect from cloud.
If user don't know how to use one command, he can also learn the command detail from the rest endpoint description.





Advantages

1. For new user, allow he run or use software on cloud without login that VM or know more detail about the software commands.

2. Allow cloud operator only exposes approved commands to end-user when using VM or container, without set complex security control to limit service/command usage.

3. It's faster if there are many different software services are used in cloud, user can run or use softwares in one place.

Prior art search and results comparison

Title	Publication Number	Publication Date	Details.
COMMON MESSAGE HEADER BEARER METHOD AND DEVICE FOR CONVERTING SIMPLE OBJECT ACCESS PROTOCOL APPLICATION PROGRAMMING INTERFACE (SOAP API) INTO REPRESENTIONAL STATE TRANSFER APPLICATION PROGRAMMING INTERFACE (REST API)	WO2011150818A1	Dec 2011	The patent describes a system that convert SOAP API to REST API, it is not related to our invention.
Api Server	US20160080493A1	March 2016	This patent describes a system that can extract data from relational database and put them into no-SQL database(stored as JSON object) and those data can be further feed to an API server to be used by users. It is not related to our invention.
System and method for invoking application commands with web service calls	US9026587B2	Oct, 2012	This patent describes a system that allow user to invoke a console command by web-service. It is different with our disclosure, while ours are trying to discover them.

All of these terms: ("convert" AND "swagger" AND "rest api" AND ("shell scripts" OR "shell commands" OR "command-line" OR "shell command")) Any of these terms: "convert to "[H], "api"[M], "application programming interface"[L], "application programming"[L], "programming interface"[L], "appropriate apis"[L], "command shell"[L], "remote api"[L], "proprietary api"[L], "part of the api"[L], "same apis"[L], "native api"[L], "cross-platform"[L]

Business case and value proposition

1. Our IBM partners and competitors or softwared providers can use this invention to enhance their services.

IBM can use this invention to enhance the hybird cloud integration.

Discoverability

It can be easily discovered if software provider put their software service on cloud and expose as REST API.

It can be easily discovered if new software commands are updated from prebuild stage.