Full examples of iControlREST for device and application service deployment

ChrisMutzel, 2015-09-11

In a previous article, I highlighted a proof-of-concept where we fully automated the deployment of BIG-IP in AWS using the web interfaces of BIG-IP inconjunction with Ansible. The goal of this article is to focus in more detail on the use of iControlREST within that project, in order to show how it can be extremely useful for automating various aspects of your ITOM workflows.

There are four main workflows we execute in order to configure BIG-IP in AWS and to deploy services. This breakown is provided below.

- Basic System Configuration
- AWS-specific system configuration
- Network attachment
- Application service provisioning

For now, we avoid the discussion about which configuration elements are part of the infrastructure deployment or the application deployment. This is an important discussion, but one that can only take place after we understand how to provision the elements that will be a part of either workflow.

For each of these workflows, we have provided the log output which shows REST calls and responses. To gain from these examples, it is important to understand the following:

- These iControlREST calls were scraped from an execution of the aws-deployments code where we captured log output. In that project, we have written a custom Ansible module called 'bigip_config' (see /library/bigip_config.py in the project directory of Github). This module is used to provision config objects within TMOS using iControlREST.
- In order to make it easier to use this 'bigip_config' Ansible module, we aimed to make it idempotent. This means that we need only identify the resources we wish to create or update, and identify the state of the resource after our module is run. We don't need to worry about whether the object exists when calling our module, or the procedural set of calls that should be made it order to get it there. An example might be like: "create iApp service 'my_app' with parameters X, Y, and Z". We don't care whether 'my_app' already exists. In order to implement such behavior, the module internally does an HTTP GET against the resource or collection it is modifying. Subsequently, if the resource already exists, a PATCH call is made, otherwise a POST call is made.
- In many cases in the code, we repeat a call until it returns successfully or returns the state we are expecting. You can see this in the examples, where the same call seems to be made repeatedly.

Basic system configuration

- The examples below show how we are configuring basic device settings with REST.
- First, because BIG-IP was just started, we wait until the first iCR call, "GET mgmt/tm/sys/db", succeeds before we continue.
- The final step the workflow involves provisioning modules on BIG-IP. Note the 30 second wait between provisioning of AVR and ASM reflected in the timestamps.

```
2015-11-12 09:46:03 : Disabling Setup Utility in GUI
GET mgmt/tm/sys/db ""
2015-11-12 09:46:10 : Disabling Setup Utility in GUI
GET mgmt/tm/sys/db ""
2015-11-12 09:46:17 : Disabling Setup Utility in GUI
```

```
GET mgmt/tm/sys/db ""
Method GET mgmt/tm/sys/db returned: {"kind":"tm:sys:db:dbcollectionstate","selfLink":"https://localho

2015-11-12 09:46:19 : Disabling Setup Utility in GUI
PATCH mgmt/tm/sys/db/setup.run?ver=11.6.0 {"value": "false"}
Method PATCH mgmt/tm/sys/db/setup.run?ver=11.6.0 returned: {"kind":"tm:sys:db:dbstate","name":"setup.

2015-11-12 09:46:22 : Configuring NTP servers
PATCH mgmt/tm/sys/ntp {"timezone": "America/Los_Angeles", "servers": ["0.pool.ntp.org", "1.pool.ntp.o
Method PATCH mgmt/tm/sys/ntp returned: {"kind":"tm:sys:ntp:ntpstate","selfLink":"https://localhost/mg

2015-11-12 09:46:25 : Configuring syslog logging destinations
PATCH mgmt/tm/sys/syslog {"include": "destination loghost { udp( 10.0.3.32 port (514));};"}
Method PATCH mgmt/tm/sys/syslog returned: {"kind":"tm:sys:syslog:syslogstate","selfLink":"https://loc

2015-11-12 09:46:28 : Configuring HTTP mgmt access
PATCH mgmt/tm/sys/httpd {"allow": ["ALL"]}
Method PATCH mgmt/tm/sys/httpd returned: {"kind":"tm:sys:httpd:httpdstate","selfLink":"https://localh

2015-11-12 09:46:31 : Configuring SSH mgmt access
PATCH mgmt/tm/sys/sshd {"allow": ["ALL"]}
Method PATCH mgmt/tm/sys/sshd returned: {"kind":"tm:sys:sshd:sshdstate","selfLink":"https://localhost

2015-11-12 09:46:33 : Configuring SNMP access
PATCH mgmt/tm/sys/snmp {"allowedAddresses": ["172.16.0.0/16"]}
Method PATCH mgmt/tm/sys/snmp returned: {"kind":"tm:sys:snmp:snmpstate","selfLink":"https://localhost

2015-11-12 09:46:36 : Configuring FastL4 profiles ... fastL4-route-friendly
GET mgmt/tm/ltm/profile/fastl4 ""
Method GET mgmt/tm/ltm/profile/fastl4 returned: {"kind":"tm:ltm:profile:fastl4:fastl4collectionstate"

2015-11-12 09:46:37 : Configuring FastL4 profiles ... fastL4-route-friendly
POST mgmt/tm/ltm/profile/fastl4 {"looseClose": "enabled", "resetOnTimeout": "disabled", "name": "fast
Method POST mgmt/tm/ltm/profile/fastl4 returned: {"kind":"tm:ltm:profile:fastl4:fastl4state","name":"

2015-11-12 09:46:39 : Configuring TCP profiles ... ssl-wan-optimized
GET mgmt/tm/ltm/profile/tcp ""
Method GET mgmt/tm/ltm/profile/tcp returned: {"kind":"tm:ltm:profile:tcp:tcpcollectionstate","selfLin

2015-11-12 09:46:39 : Configuring TCP profiles ... ssl-wan-optimized
POST mgmt/tm/ltm/profile/tcp {"ackOnPush": "disabled", "nagle": "disabled", "delayedAcks": "disabled"
Method POST mgmt/tm/ltm/profile/tcp returned: {"kind":"tm:ltm:profile:tcp:tcpstate","name":"tcp-ssl-w

2015-11-12 09:46:41 : Configuring TCP profiles ... ssl-lan-optimized
GET mgmt/tm/ltm/profile/tcp ""
Method GET mgmt/tm/ltm/profile/tcp returned: {"kind":"tm:ltm:profile:tcp:tcpcollectionstate","selfLin

2015-11-12 09:46:42 : Configuring TCP profiles ... ssl-lan-optimized
POST mgmt/tm/ltm/profile/tcp {"ackOnPush": "disabled", "nagle": "disabled", "delayedAcks": "disabled"
Method POST mgmt/tm/ltm/profile/tcp returned: {"kind":"tm:ltm:profile:tcp:tcpstate","name":"tcp-ssl-l

2015-11-12 09:46:44 :
PATCH mgmt/tm/sys/provision/asm {"level": "nominal"}
Method PATCH mgmt/tm/sys/provision/asm returned: {"kind":"tm:sys:provision:provisionstate","name":"as

2015-11-12 09:47:18 :
PATCH mgmt/tm/sys/provision/avr {"level": "nominal"}
Method PATCH mgmt/tm/sys/provision/avr returned: {"kind":"tm:sys:provision:provisionstate","name":"av
```

AWS-specific System Configuration

- This next workflow is very small and simple. We are adding some variables to global-settings which are only necessary because BIG-IP is running in AWS.
- We've obfuscated the AWS Access Key and Secret Key in the output.

```
2015-11-12 09:48:12 : Adding/updating AWS access and secret keys
PATCH mgmt/tm/sys/global-settings {"awsAccessKey": "...<my access key>...", "awsSecretKey": "...<my s
Method PATCH mgmt/tm/sys/global-settings returned: {"kind":"tm:sys:global-settings:global-settingssta
```

Network Attachment

- Setup of self-IPs, VLANs, and other network specific configuration is relatively straight forward.

```
2015-11-12 09:48:15 : Disabling dhcp
PATCH mgmt/tm/sys/db/dhclient.mgmt {"value": "disable"}
Method PATCH mgmt/tm/sys/db/dhclient.mgmt returned: {"kind":"tm:sys:db:dbstate","name":"dhclient.mgmt

2015-11-12 09:48:18 : Adding/updating internal vlan
GET mgmt/tm/net/vlan ""
Method GET mgmt/tm/net/vlan returned: {"kind":"tm:net:vlan:vlancollectionstate","selfLink":"https://l

2015-11-12 09:48:19 : Adding/updating internal vlan
POST mgmt/tm/net/vlan {"interfaces": "1.2", "name": "private"}
Method POST mgmt/tm/net/vlan returned: {"kind":"tm:net:vlan:vlanstate","name":"private","fullPath":"p

2015-11-12 09:48:21 : Adding/updating external vlan
GET mgmt/tm/net/vlan ""
Method GET mgmt/tm/net/vlan returned: {"kind":"tm:net:vlan:vlancollectionstate","selfLink":"https://l

2015-11-12 09:48:22 : Adding/updating external vlan
POST mgmt/tm/net/vlan {"interfaces": "1.1", "name": "public"}
Method POST mgmt/tm/net/vlan returned: {"kind":"tm:net:vlan:vlanstate","name":"public","fullPath":"pu

2015-11-12 09:48:24 : Adding/updating internal selfip
GET mgmt/tm/net/self ""
Method GET mgmt/tm/net/self returned: {"kind":"tm:net:self:selfcollectionstate","selfLink":"https://l

2015-11-12 09:48:24 : Adding/updating internal selfip
POST mgmt/tm/net/self {"allowService": "default", "vlan": "private", "trafficGroup": "traffic-group-l
Method POST mgmt/tm/net/self returned: {"kind":"tm:net:self:selfstate","name":"private","fullPath":"p

2015-11-12 09:48:26 : Adding/updating external selfip
GET mgmt/tm/net/self ""
Method GET mgmt/tm/net/self returned: {"kind":"tm:net:self:selfcollectionstate","selfLink":"https://l

2015-11-12 09:48:27 : Adding/updating external selfip
POST mgmt/tm/net/self {"allowService": ["tcp:4353"], "vlan": "public", "trafficGroup": "traffic-group
Method POST mgmt/tm/net/self returned: {"kind":"tm:net:self:selfstate","name":"public","fullPath":"pu

2015-11-12 09:48:29 : Setting default route using default_gateway or gateway_pool
GET /mgmt/tm/net/route ""
Method GET /mgmt/tm/net/route returned: {"kind":"tm:net:route:routecollectionstate","selfLink":"https
```

```
2015-11-12 09:48:29 : Setting default route using default_gateway or gateway_pool
POST /mgmt/tm/net/route {"gw": "172.16.13.1", "name": "default_route", "network": "default"}
Method POST /mgmt/tm/net/route returned: {"kind":"tm:net:route:routestate","name":"default_route","fu
```

Application Service Provisioning

This workflow is where things really get interesting. Let's break it down.

- We are deploying two sets of virtual servers (the pool members are the same, but the VIP is different).
- For virtual 1 (VIP = 172.16.13.128), we use an iApp to deploy a HTTPS virtual with an ASM policy. To do so, we:
    - Deploy all resources that are needed to support the iApp deployment:
        - A high-speed logging pool
        - An LTM logging profile (which will send logs to Splunk on port 514)
        - An ASM logging profile (which will send logs to Splunk on port 515)
        - An analytics profile, in case we want to inspect traffic with AVR on-box
        - Base64 encoded images to an iRule data-group
        - iRules to support a sorry page and the analytics profile
        - An ASM policy (we've encoded the XML policy file into base64). Deploying the ASM policies requires first making a new policy via a POST command, then importing the policy over the defaults for the one we have just created.
            - Note that we check the status of the asynchronous REST tasks which are started during the policy 'create' and 'apply' steps.
        - An LTM policy which attaches the ASM policy above using a ruleset.
    - Deploy the iApp template (look here to understand how we built the JSON payload for the iApp template).
    - Finally, deploy the iApp service, an instantiation of the template that references all the above content (look here to understand how we built the JSON payload for the iApp service).
- For virtual 2 (VIP = 172.16.13.124), just deploy the web server pool, iRule, and virtual server directly (without an iApp).

```
2015-11-12 09:49:13 : Deploying/updating Webserver Pool
GET mgmt/tm/ltm/pool ""
Method GET mgmt/tm/ltm/pool returned: {"kind":"tm:ltm:pool:poolcollectionstate","selfLink":"https://l

2015-11-12 09:49:14 : Deploying/updating Webserver Pool
POST mgmt/tm/ltm/pool {"name": "Vip1_pool", "members": [{"description": "Name=/boring_lovelace,Contai
Method POST mgmt/tm/ltm/pool returned: {"kind":"tm:ltm:pool:poolstate","name":"Vip1_pool","fullPath":

2015-11-12 09:49:18 : Deploying/updating High Speed Logging pool to send to Analytics Server
GET mgmt/tm/ltm/pool ""
Method GET mgmt/tm/ltm/pool returned: {"kind":"tm:ltm:pool:poolcollectionstate","selfLink":"https://l

2015-11-12 09:49:19 : Deploying/updating High Speed Logging pool to send to Analytics Server
POST mgmt/tm/ltm/pool {"name": "syslog_pool", "members": [{"name": "172.16.14.180:514", "address": "1
Method POST mgmt/tm/ltm/pool returned: {"kind":"tm:ltm:pool:poolstate","name":"syslog_pool","fullPath

2015-11-12 09:49:21 : Deploying/updating ASM Logging Profile to send to Remote Analytics Server
GET mgmt/tm/security/log/profile ""
Method GET mgmt/tm/security/log/profile returned: {"kind":"tm:security:log:profile:profilecollections

2015-11-12 09:49:21 : Deploying/updating ASM Logging Profile to send to Remote Analytics Server
POST mgmt/tm/security/log/profile {"application": [{"guaranteeLogging": "enabled", "guaranteeResponse
Method POST mgmt/tm/security/log/profile returned: {"kind":"tm:security:log:profile:profilestate","na

2015-11-12 09:49:23 : Deploying/updating Analytics Profile
```

```
2015-11-12 09:49:23 : Deploying/updating Analytics Profile
GET mgmt/tm/ltm/profile/analytics ""
Method GET mgmt/tm/ltm/profile/analytics returned: {"kind":"tm:ltm:profile:analytics:analyticscollect

2015-11-12 09:49:24 : Deploying/updating Analytics Profile
POST mgmt/tm/ltm/profile/analytics {"collectPageLoadTime": "enabled", "notificationBySnmp": "disabled
Method POST mgmt/tm/ltm/profile/analytics returned: {"kind":"tm:ltm:profile:analytics:analyticsstate"

2015-11-12 09:49:26 : Uploading Datagroup ... background for sorry page
GET mgmt/tm/ltm/data-group/internal ""
Method GET mgmt/tm/ltm/data-group/internal returned: {"kind":"tm:ltm:data-group:internal:internalcoll

2015-11-12 09:49:26 : Uploading Datagroup ... background for sorry page
POST mgmt/tm/ltm/data-group/internal {"records": [{"name": "...<base64 image>..."}], "type": "string"
Method POST mgmt/tm/ltm/data-group/internal returned: {"kind":"tm:ltm:data-group:internal:internalsta

2015-11-12 09:49:29 : Uploading Datagroup ... image for sorry page
GET mgmt/tm/ltm/data-group/internal ""
Method GET mgmt/tm/ltm/data-group/internal returned: {"kind":"tm:ltm:data-group:internal:internalcoll

2015-11-12 09:49:30 : Uploading Datagroup ... image for sorry page
POST mgmt/tm/ltm/data-group/internal {"records": [{"name": "...<base64 image>..."}], "type": "string"
Method POST mgmt/tm/ltm/data-group/internal returned: {"kind":"tm:ltm:data-group:internal:internalsta

2015-11-12 09:49:32 : Uploading iRules ... sorry_page_rule
GET mgmt/tm/ltm/rule ""
Method GET mgmt/tm/ltm/rule returned: {"kind":"tm:ltm:rule:rulecollectionstate","selfLink":"https://l

2015-11-12 09:49:33 : Uploading iRules ... sorry_page_rule
POST mgmt/tm/ltm/rule {"apiAnonymous": "when HTTP_REQUEST {\n  set VSPool [LB::server pool]\n  if { [
Method POST mgmt/tm/ltm/rule returned: {"kind":"tm:ltm:rule:rulestate","name":"irule_sorry_page","ful

2015-11-12 09:49:35 : Uploading iRules ... demo_analytics_rule
GET mgmt/tm/ltm/rule ""
Method GET mgmt/tm/ltm/rule returned: {"kind":"tm:ltm:rule:rulecollectionstate","selfLink":"https://l

2015-11-12 09:49:36 : Uploading iRules ... demo_analytics_rule
POST mgmt/tm/ltm/rule {"apiAnonymous": "when CLIENT_ACCEPTED {\n    set client [IP::client_addr]\n}\n
Method POST mgmt/tm/ltm/rule returned: {"kind":"tm:ltm:rule:rulestate","name":"irule_demo_analytics",

2015-11-12 09:49:38 : Create the ASM policy
GET mgmt/tm/asm/policies ""
Method GET mgmt/tm/asm/policies returned: {"selfLink":"https://localhost/mgmt/tm/asm/policies","kind"

2015-11-12 09:49:39 : Create the ASM policy
POST mgmt/tm/asm/policies {"caseInsensitive": true, "name": "linux_high-Vip1", "applicationLanguage":
Method POST mgmt/tm/asm/policies returned: {"historyRevisionReference":{"link":"https://localhost/mgm

2015-11-12 09:49:53 : Import our policy over the one existing above
POST mgmt/tm/asm/tasks/import-policy {"policyReference": {"link": "https://localhost/mgmt/tm/asm/poli

2015-11-12 09:50:00 : Determine whether the asm policy import task is complete
GET mgmt/tm/asm/tasks/import-policy/hP37L9EM650WeWKkgX7law ""
Method GET mgmt/tm/asm/tasks/import-policy/hP37L9EM650WeWKkgX7law returned: {"isBase64":true,"status"

2015-11-12 09:50:05 : Determine whether the asm policy import task is complete
GET mgmt/tm/asm/tasks/import-policy/hP37L9EM650WeWKkgX7law ""
Method GET mgmt/tm/asm/tasks/import-policy/hP37L9EM650WeWKkgX7law returned: {"isBase64":true,"status"
```

```
2015-11-12 09:50:08 : Apply the ASM policy
POST mgmt/tm/asm/tasks/apply-policy {"policyReference": {"link": "https://localhost/mgmt/tm/asm/polic
Method POST mgmt/tm/asm/tasks/apply-policy returned: {"selfLink":"https://localhost/mgmt/tm/asm/tasks

2015-11-12 09:50:10 : Determine whether the asm policy apply task is complete
GET mgmt/tm/asm/tasks/apply-policy/38B8slfPm1_lBBRG1STNeg ""
Method GET mgmt/tm/asm/tasks/apply-policy/38B8slfPm1_lBBRG1STNeg returned: {"selfLink":"https://local

2015-11-12 09:50:14 : Determine whether the asm policy apply task is complete
GET mgmt/tm/asm/tasks/apply-policy/38B8slfPm1_lBBRG1STNeg ""
Method GET mgmt/tm/asm/tasks/apply-policy/38B8slfPm1_lBBRG1STNeg returned: {"selfLink":"https://local

2015-11-12 09:50:18 : Determine whether the asm policy apply task is complete
GET mgmt/tm/asm/tasks/apply-policy/38B8slfPm1_lBBRG1STNeg ""
Method GET mgmt/tm/asm/tasks/apply-policy/38B8slfPm1_lBBRG1STNeg returned: {"status":"COMPLETED","las

2015-11-12 09:50:20 : Create an LTM policy for use with by iApp which associates the ASM policy
GET mgmt/tm/ltm/policy ""
Method GET mgmt/tm/ltm/policy returned: {"kind":"tm:ltm:policy:policycollectionstate","selfLink":"htt

2015-11-12 09:50:23 : Create an LTM policy for use with by iApp which associates the ASM policy
POST mgmt/tm/ltm/policy {"name": "ltm_policy_w_asm_linux_high-Vip1", "rules": [{"ordinal": 1, "condit
Method POST mgmt/tm/ltm/policy returned: {"kind":"tm:ltm:policy:policystate","name":"ltm_policy_w_asm

2015-11-12 09:50:25 : Deploy the iApp template, since we are not using a default iApp on box
GET mgmt/tm/sys/application/template ""
Method GET mgmt/tm/sys/application/template returned: {"kind":"tm:sys:application:template:templateco

2015-11-12 09:50:26 : Deploy the iApp template, since we are not using a default iApp on box
POST mgmt/tm/sys/application/template {"name": "f5.http.backport.1.1.2", "actions": [{"implementation
Method POST mgmt/tm/sys/application/template returned: {"kind":"tm:sys:application:template:templates

2015-11-12 09:50:30 : Deploy the iApp service from the f5 http backport template
GET mgmt/tm/sys/application/service ""
Method GET mgmt/tm/sys/application/service returned: {"kind":"tm:sys:application:service:servicecolle

2015-11-12 09:50:30 : Deploy the iApp service from the f5 http backport template
POST mgmt/tm/sys/application/service {"tables": [{"name": "basic__snatpool_members"}, {"name": "net__
Method POST mgmt/tm/sys/application/service returned: {"kind":"tm:sys:application:service:servicestat

2015-11-12 09:51:37 : Deploying/updating webserver pool
GET mgmt/tm/ltm/pool ""
Method GET mgmt/tm/ltm/pool returned: {"kind":"tm:ltm:pool:poolcollectionstate","selfLink":"https://l

2015-11-12 09:51:37 : Deploying/updating webserver pool
POST mgmt/tm/ltm/pool {"name": "Vip2_pool", "members": [{"description": "Name=/boring_lovelace,Contai
Method POST mgmt/tm/ltm/pool returned: {"kind":"tm:ltm:pool:poolstate","name":"Vip2_pool","fullPath":

2015-11-12 09:51:39 : Uploading iRules ... irule_random_snat
GET mgmt/tm/ltm/rule ""
Method GET mgmt/tm/ltm/rule returned: {"kind":"tm:ltm:rule:rulecollectionstate","selfLink":"https://l

2015-11-12 09:51:40 : Uploading iRules ... irule_random_snat
POST mgmt/tm/ltm/rule {"apiAnonymous": "when RULE_INIT {\n    expr srand(\"[clock clicks]\")\n    set
Method POST mgmt/tm/ltm/rule returned: {"kind":"tm:ltm:rule:rulestate","name":"irule_random_snat","fu

2015-11-12 09:51:42 : Setup the HTTP virtual server
```

```
2015-11-12 09:51:42 : Setup the HTTP virtual server
GET mgmt/tm/ltm/virtual ""
Method GET mgmt/tm/ltm/virtual returned: {"kind":"tm:ltm:virtual:virtualcollectionstate","selfLink":"

2015-11-12 09:51:43 : Setup the HTTP virtual server
POST mgmt/tm/ltm/virtual {"name": "Vip2_http", "rules": ["/Common/irule_random_snat"], "translateAddr
Method POST mgmt/tm/ltm/virtual returned: {"kind":"tm:ltm:virtual:virtualstate","name":"Vip2_http","f

2015-11-12 09:51:45 : Setup the HTTPS virtual server
GET mgmt/tm/ltm/virtual ""
Method GET mgmt/tm/ltm/virtual returned: {"kind":"tm:ltm:virtual:virtualcollectionstate","selfLink":"

2015-11-12 09:51:45 : Setup the HTTPS virtual server
POST mgmt/tm/ltm/virtual {"name": "Vip2_https", "rules": ["/Common/irule_random_snat"], "translateAdd
Method POST mgmt/tm/ltm/virtual returned: {"kind":"tm:ltm:virtual:virtualstate","name":"Vip2_https","

2015-11-12 09:51:47 :
GET mgmt/tm/ltm/virtual ""
Method GET mgmt/tm/ltm/virtual returned: {"kind":"tm:ltm:virtual:virtualcollectionstate","selfLink":"
```

Deploying an iApp template using iControlREST

Because we recognize that it may be not obvious how we are deploying iApp templates using iControlREST, we break it down into more detail here.

First, note that there is no 'import' action we can invoke via REST to import the iApp template which mirrors the action in the Configuration Utility (GUI). This means that we need to create the JSON payload containing the iApp and POST it.

Given an iApp template, like those found on DevCentral, here are the steps to create the JSON body.

1. On a pre-existing BIG-IP install (or one have created in your build process for your code)
    1. Import the iApp template in the Configuration Utility in the 'Common' partition
    2. Do an HTTP GET to retrieve the iApp template payload. Make sure that you use the expandSubcollections=True as a query parameter, as we want to include the stuff in the 'actionsReference' sub-collection.

        ```
        curl -sku <user>:<password> -X GET  https://<management ip>/mgmt/tm/sys/application/te
        ```

    3. You should get something back that looks like the following (which is the payload for the f5.http backport iApp). I have truncated the 'implementation', 'presentation' and 'htmlHelp' actions:

        ```
        {
            "actionsReference": {
                "isSubcollection": true,
                "items": [
                    {
                        "fullPath": "definition",
                        "generation": 4672,
                        "htmlHelp": "....",
                        "implementation": "...",
                        "kind": "tm:sys:application:template:actions:actionsstate",
                        "name": "definition",
        ```

```
                "presentation": "...",
                "roleAcl": [
                    "admin",
                    "manager",
                    "resource-admin"
                ],
                "selfLink": "https://localhost/mgmt/tm/sys/application/template/~Commo
            }
        ],
        "link": "https://localhost/mgmt/tm/sys/application/template/~Common~f5.http.ba
    },
    "fullPath": "/Common/f5.http.backport.1.1.2",
    "generation": 4672,
    "ignoreVerification": "false",
    "kind": "tm:sys:application:template:templatestate",
    "name": "f5.http.backport.1.1.2",
    "partition": "Common",
    "requiresBigipVersionMin": "11.6.0",
    "selfLink": "https://localhost/mgmt/tm/sys/application/template/~Common~f5.http.ba
    "totalSigningStatus": "not-all-signed",
    "verificationStatus": "none"
}
```


2. Before we can POST this payload back to any BIG-IP, we need to cleanup a few things:
    1. Remove any of the extraneous fields including 'verificationStatus', 'totalSigningStatus', 'selfLink', 'partition', 'kind', 'generation', 'fullPath'.
    2. Make a new top-level key in the payload called 'actions'. The value for this key should everything in the 'items' array under the top-level key 'actionsReference'. Finally, delete the 'actionsReference' key/value pair from the JSON body. The final JSON payload should look like:

```
{
 "actions": [
   {
     "htmlHelp": "....",
     "implementation": "...",
     "name": "definition",
     "presentation": "...",
     "roleAcl": [
       "admin",
       "manager",
       "resource-admin"
     ]
   }
 ],
 "ignoreVerification": "false",
 "name": "f5.http.backport.1.1.2",
 "requiresBigipVersionMin": "11.6.0",
 "totalSigningStatus": "not-all-signed"
}
```

3. Finally, we can use this to deploy an iApp template on BIG-IP. In the example below, the iApp_template.json file is formatted like the above. I have also **attached it to this page** for inspection.

```
curl -sku rest_admin:<obfuscated> -H "Content-type: application/json" -X POST -d@
```

Before you go POSTing iApps to any old version of TMOS, be aware that there are still some remaining issues you might have to solve. Some of the official iApps found on DevCentral are prepended with a TCL library that defines functions used within the iApp. The iApp solutions team made this design decision so that newer iApps will work against older versions of TMOS. For example, see the 'F5 HTTP', which starts with the library definition on line 0: "cli script f5.iapp.1.3.0.cli {....". When you export the iApp template to JSON using REST as we have documented above, this library will not be included in the payload. Because newer versions of BIG-IP (11.6) might already include a version of this 'iApp' library, you can work around this issue by updating the function references to use the existing library on-box. Here are the high-level steps:

1. Downloading the iApp from DevCentral
2. Change the function references to leverage the library that is installed on your BIG-IP. See an example of this by comparing the F5 HTTP template on the codeshare with the one attached to this page.
    1. You'll probably have to do some "find and replace" like the following:

```
f5.iapp.1.3.0.cli:iapp_get_provisioned -> iapp::get_provisioned
```

    3. There may be some references to functions that do not exist yet. These will have to be dealt with on a case-by-case basis.
3. Uploading the iApp to BIG-IP and exporting we documented above.

Deploying an iApp service using iControlREST

Fortunately, using iControlREST to manage instances of iApps (also known as iApp services) is much easier than managing templates. The high-level steps are similar:

1. Deploy an iApp service via the Configuration Utility.
2. Do an HTTP GET to acquire the JSON representation (notice the URL formatting!).

```
curl -sku <user>:<password> -X GET https://<management ip>/mgmt/tm/sys/application/service/
```

3. Depending on the variables presented by the iApp template, the JSON payload for the iApp service might look something like:

```
{
   "deviceGroup": "none",
   "fullPath": "/Common/Vip1_iApp.app/Vip1_iApp",
   "generation": 4674,
   "inheritedDevicegroup": "true",
   "inheritedTrafficGroup": "true",
   "kind": "tm:sys:application:service:servicestate",
   "lists": [
       {
           "encrypted": "no",
           "name": "irules__irules",
           "value": [
               "/Common/irule_demo_analytics",
               "/Common/irule_sorry_page"
           ]
       }
   ],
   "name": "Vip1_iApp",
   "partition": "Common",
   "selfLink": "https://localhost/mgmt/tm/sys/application/service/~Common~Vip1_iApp.app~Vi
   "strictUpdates": "enabled",
   "subPath": "Vip1_iApp.app",
   "tables": [
       {
```

```
            "name": "basic__snatpool_members"
        },
        {
            "name": "net__snatpool_members"
        },
        {
            "name": "optimizations__hosts"
        },
        {
            "columnNames": [
                "name"
            ],
            "name": "pool__hosts",
            "rows": [
                {
                    "row": [
                        "demo.example.com"
                    ]
                }
            ]
        },
        {
            "name": "pool__members"
        },
        {
            "name": "server_pools__servers"
        }
    ],
    "template": "/Common/f5.http.backport.1.1.2",
    "templateModified": "yes",
    "trafficGroup": "/Common/traffic-group-1",
    "variables": [
        {
            "encrypted": "no",
            "name": "asm__security_logging",
            "value": "asm_log_to_splunk"
        },
        {
            "encrypted": "no",
            "name": "asm__use_asm",
            "value": "/Common/ltm_policy_w_asm_linux_high-Vip1"
        },
        {
            "encrypted": "no",
            "name": "client__http_compression",
            "value": "/#do_not_use#"
        },
        {
            "encrypted": "no",
            "name": "client__standard_caching_without_wa",
            "value": "/#do_not_use#"
        },
        {
            "encrypted": "no",
            "name": "client__tcp_wan_opt",
            "value": "/Common/tcp-ssl-wan-optimized"
        },
        {
```

```
        {
            "encrypted": "no",
            "name": "net__client_mode",
            "value": "wan"
        },
        {
            "encrypted": "no",
            "name": "net__route_to_bigip",
            "value": "no"
        },
        {
            "encrypted": "no",
            "name": "net__same_subnet",
            "value": "no"
        },
        {
            "encrypted": "no",
            "name": "net__server_mode",
            "value": "lan"
        },
        {
            "encrypted": "no",
            "name": "net__snat_type",
            "value": "automap"
        },
        {
            "encrypted": "no",
            "name": "net__vlan_mode",
            "value": "all"
        },
        {
            "encrypted": "no",
            "name": "pool__addr",
            "value": "172.16.13.128"
        },
        {
            "encrypted": "no",
            "name": "pool__http",
            "value": "/#create_new#"
        },
        {
            "encrypted": "no",
            "name": "pool__mask",
            "value": "none"
        },
        {
            "encrypted": "no",
            "name": "pool__persist",
            "value": "/#cookie#"
        },
        {
            "encrypted": "no",
            "name": "pool__pool_to_use",
            "value": "/Common/Vip1_pool"
        },
        {
            "encrypted": "no",
            "name": "pool__port_secure",
```

```
            "value": "443"
        },
        {
            "encrypted": "no",
            "name": "pool__redirect_port",
            "value": "80"
        },
        {
            "encrypted": "no",
            "name": "pool__redirect_to_https",
            "value": "yes"
        },
        {
            "encrypted": "no",
            "name": "pool__xff",
            "value": "yes"
        },
        {
            "encrypted": "no",
            "name": "server__oneconnect",
            "value": "/#do_not_use#"
        },
        {
            "encrypted": "no",
            "name": "server__tcp_lan_opt",
            "value": "/Common/tcp-wan-optimized"
        },
        {
            "encrypted": "no",
            "name": "server__tcp_req_queueing",
            "value": "no"
        },
        {
            "encrypted": "no",
            "name": "ssl__cert",
            "value": "/Common/default.crt"
        },
        {
            "encrypted": "no",
            "name": "ssl__client_ssl_profile",
            "value": "/#create_new#"
        },
        {
            "encrypted": "no",
            "name": "ssl__key",
            "value": "/Common/default.key"
        },
        {
            "encrypted": "no",
            "name": "ssl__mode",
            "value": "client_ssl"
        },
        {
            "encrypted": "no",
            "name": "ssl__use_chain_cert",
            "value": "/#do_not_use#"
        },
        {
```

```
        {
            "encrypted": "no",
            "name": "ssl_encryption_questions__advanced",
            "value": "yes"
        },
        {
            "encrypted": "no",
            "name": "ssl_encryption_questions__help",
            "value": "hide"
        },
        {
            "encrypted": "no",
            "name": "stats__analytics",
            "value": "/Common/Vip1-demo_analytics"
        },
        {
            "encrypted": "no",
            "name": "stats__request_logging",
            "value": "/#do_not_use#"
        }
    ]
}
```


4. As ealier, remove some of the fields that don't make sense to re-post. This includes 'deviceGroup', 'fullPath', 'generation', 'kind', 'partition', 'selfLink', and 'subPath'.
5. You can now use this JSON body with updates to the variable values as needed.

Example python code for deploying iApp Templates

In addition to the above procedures, we'd like you point you to some python examples which show how to push iApp templates using REST. Hitesh Patel, another monster F5er, has put together the following code:

https://github.com/0xHiteshPatel/appsvcs_integration_iapp/tree/80cc40dcf85e352a25c7ec44d9e4dcc253e51e69/scripts

In his words: "that's 152 lines of awesome right there".

His examples run against 11.5.x, 11.6.x and 12.0.

Debugging

When trying to create or update an instance of an iApp via REST, you will get error messages in the HTTP response if your POST is unsuccessful. In addition to the HTTP payload in the response, the following debug steps can be helpful:

1) Set the scriptd log level to debug:

```
modify sys scriptd log-level debug
```

2) Look at the TMSH output from the iApp printed to /var/log/scriptd.out. Typically the last line will show the error that has occured.

In closing

The above examples should bring you one step closer to automating the delivery of advanced network services for your applications. We're looking forward to doing future posts on how to automate your deployment. Finally, if you haven't checked out the Application Services Integration iApp, also by Hitesh, you should probably do so now: https://github.com/0xHiteshPatel/appsvcs_integration_iapp.

Cheers!

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com