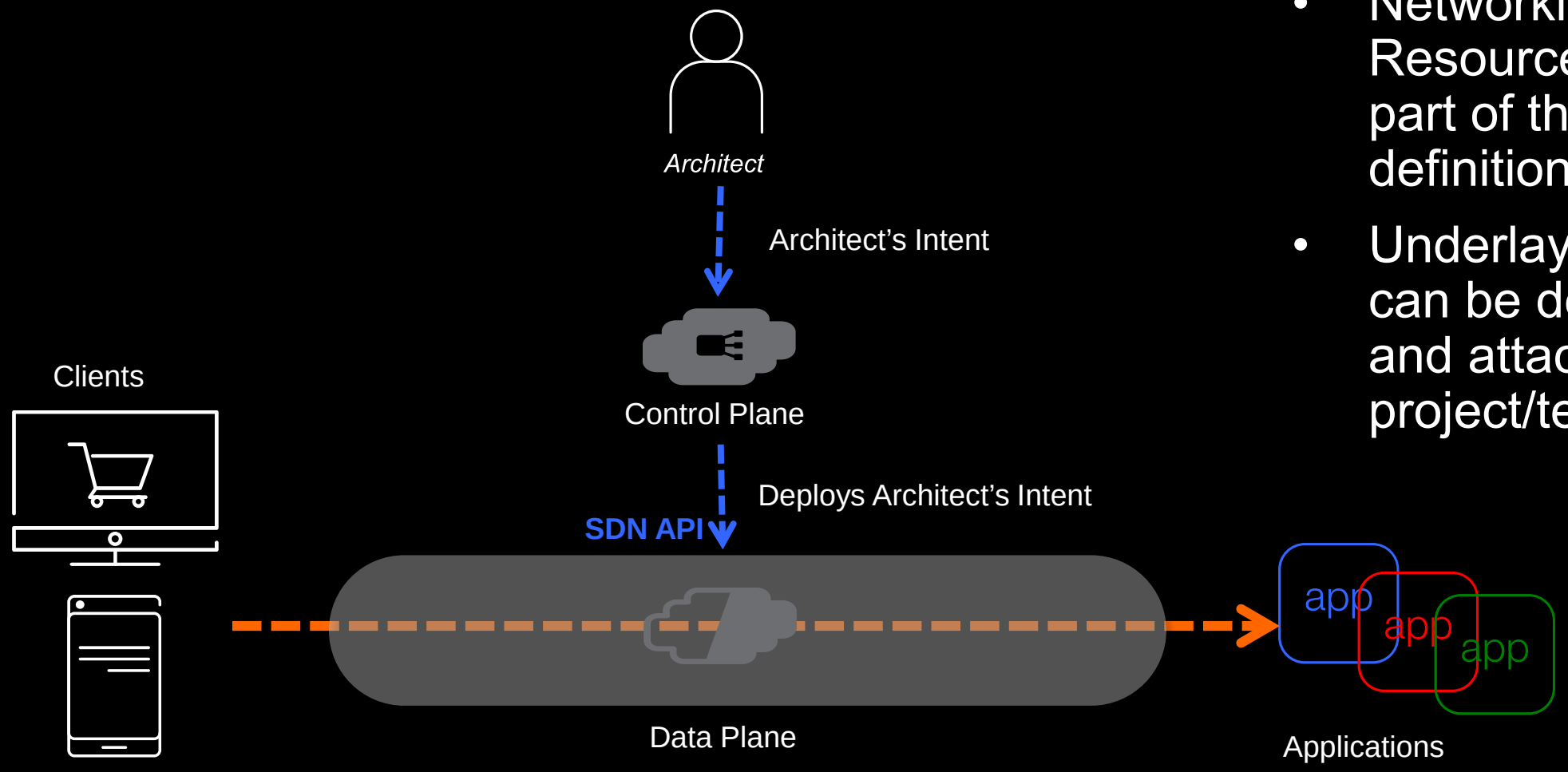
Automation with F5 and SDN Solutions

Presenter:

Philippe CLOUP,

Solution Architect EMEA, Cloud/SDN

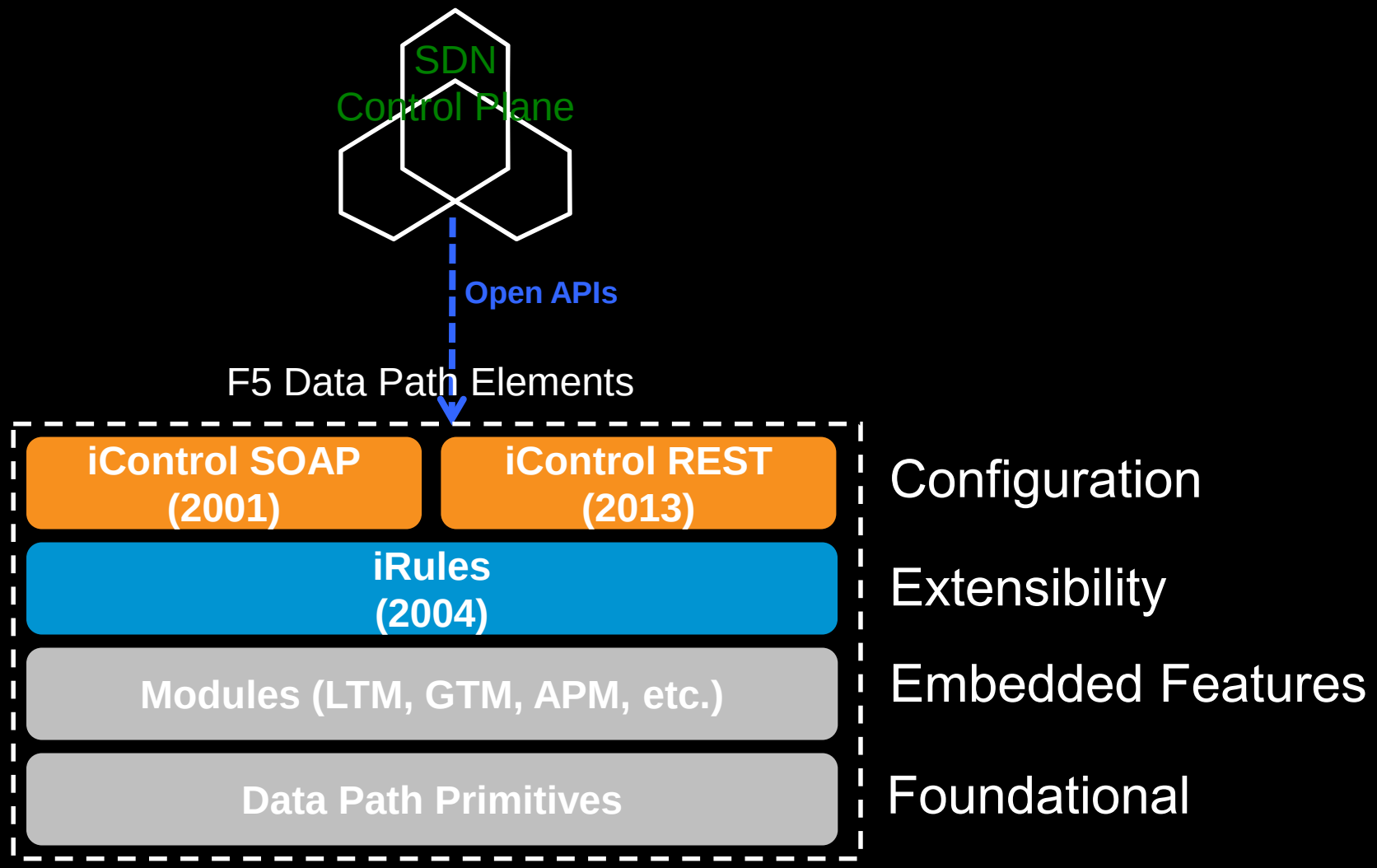
SDN – Software Defined Networking



- Networking Resources can be part of the App definition
- Underlay and Overlay can be decoupled and attached to a project/tenant

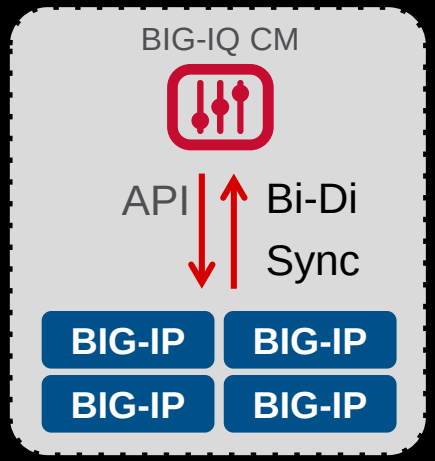
How F5 BIG-IP fits into SDN API world

Parameters from Network to Application can be configured through F5 API



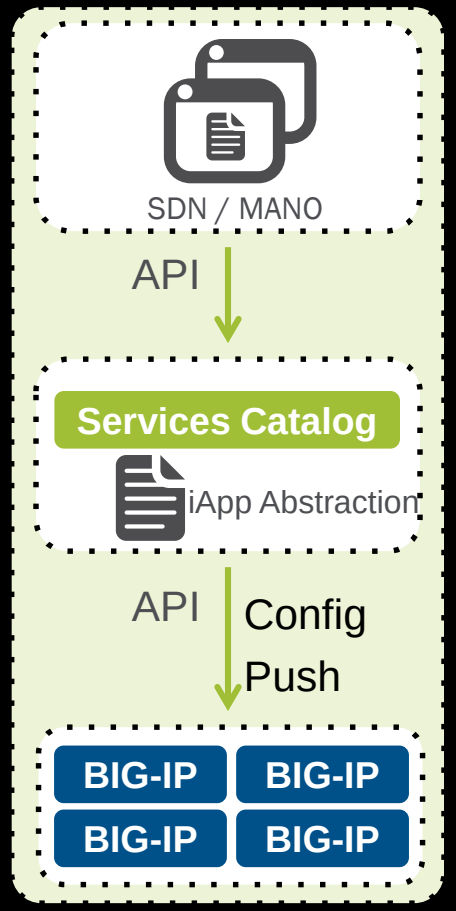
Different Orchestration Instantiations

Central Mgmt

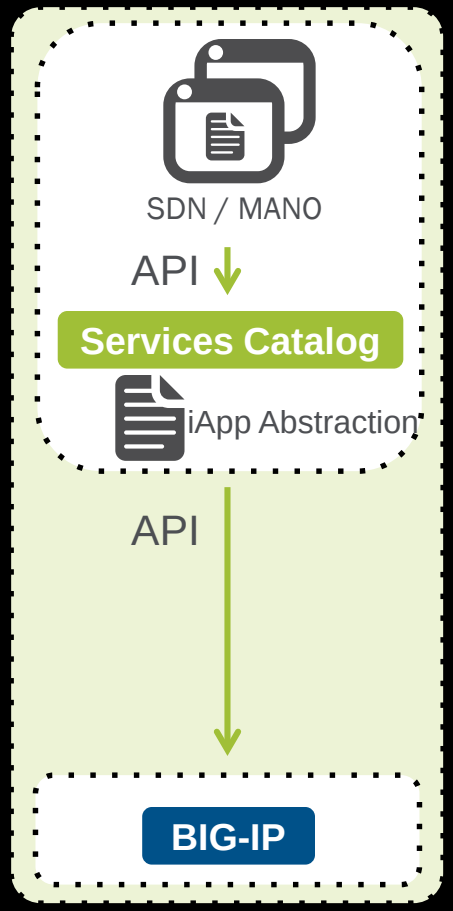


Orchestration Options

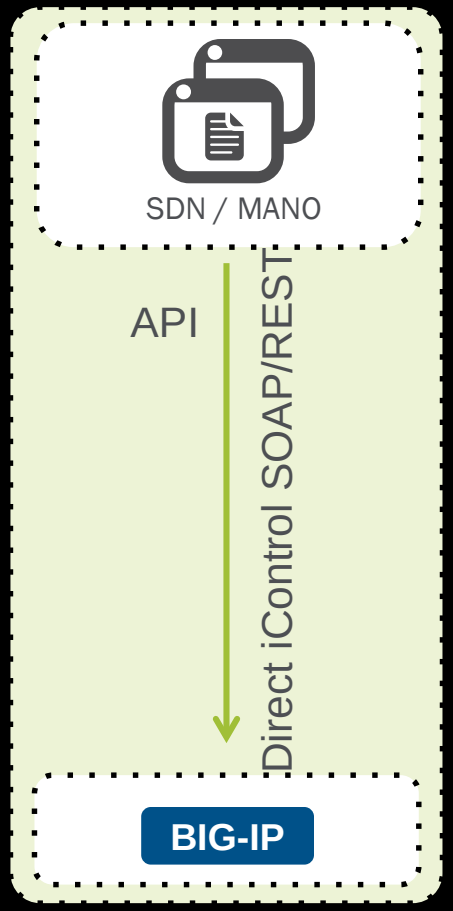
Middleware



Plugin



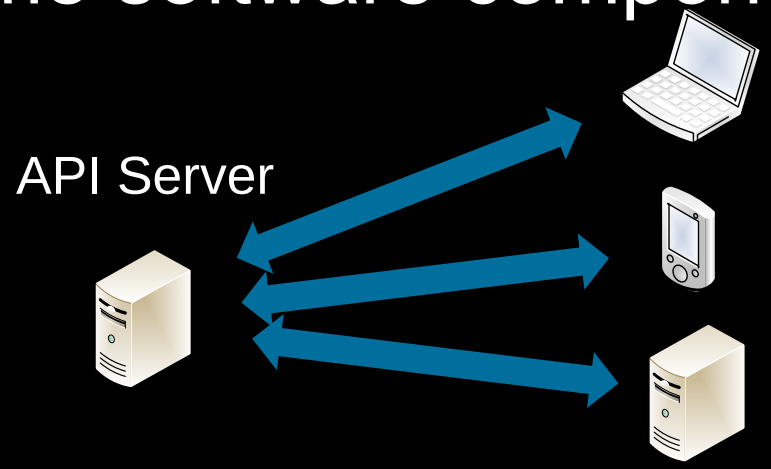
Classic



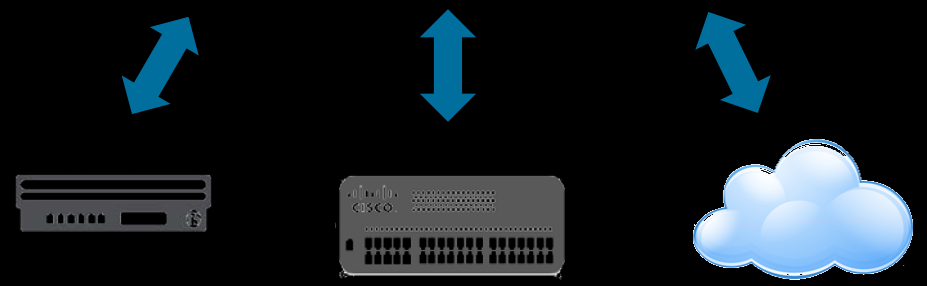
Moving into REST API

Why REST? Why Now?

An **application programming interface** (**API**) simply specifies how some software components should interact with each other

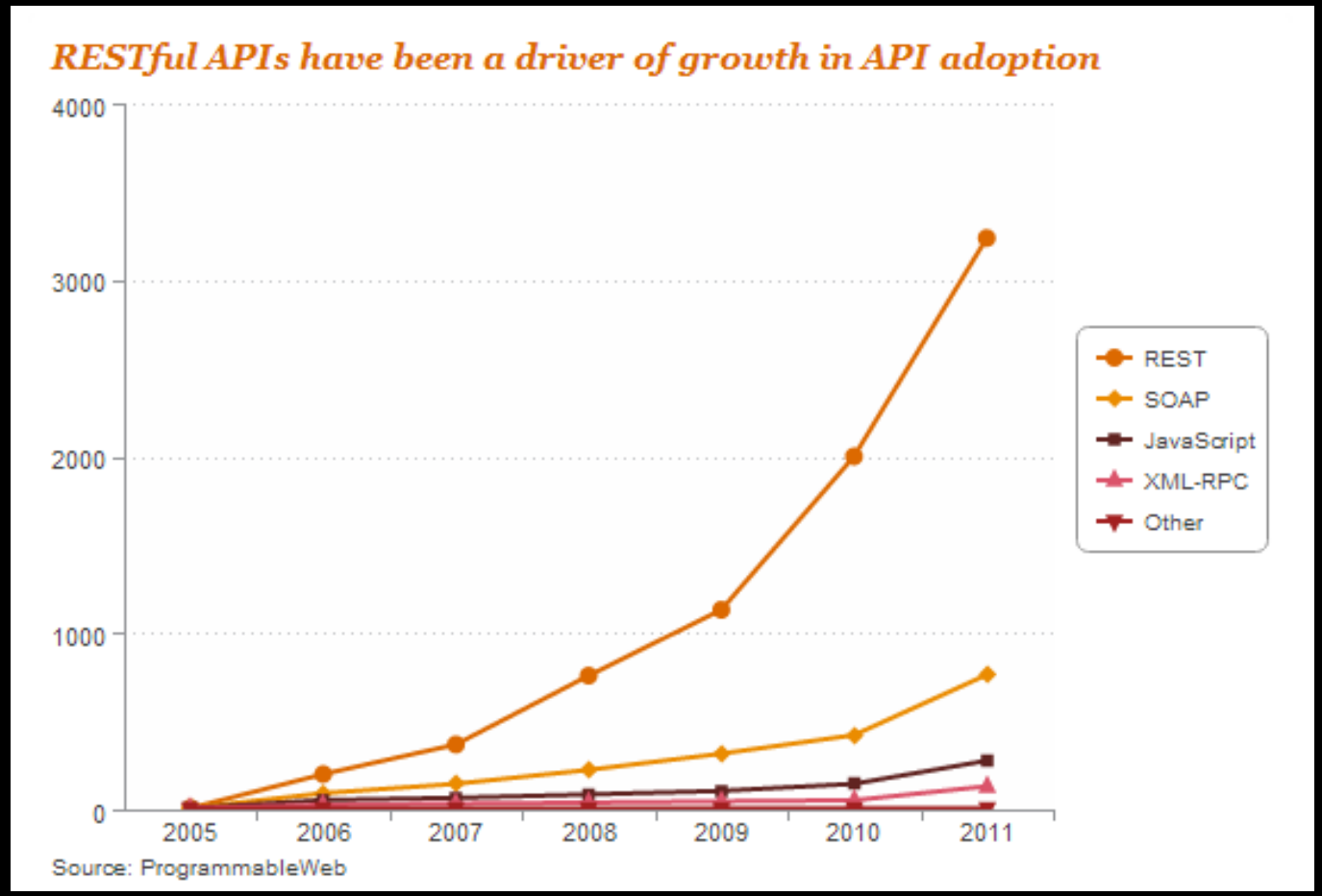


Traditional APIs were SOAP/CRUD based using XML or JSON – REST APIs are more standards based



iControl – SOAP to REST

- iControl – The original control plane automation tool from F5
  - Programmatic access to anything that you can do via the CLI or GUI
  - Remote API access
  - SOAP/XML based
- iControl REST – A new approach to remote BIG-IP scripting
  - REST based architecture uses simple, small command structures.
  - Tied directly to tmsh commands
    - Commands you know, very low bar to entry
    - Less barrier to developers promoting functionality via API
    - Symmetry between GUI/CLI & API dev/maintenance
  - Rapid development and rollout



tmsh vs iControl REST?

tmsh:

```
modify ltm pool http-pool members modify { 10.133.20.60:any { session user-enabled } }
```

iControl REST:

```
curl -k -u admin:admin -H "Content-Type: application/json" -X PUT -d '{"session": "user-enabled"}'
https://localhost/mgmt/tm/ltm/pool/http-pool/members/10.133.20.60:any
```

What’s this REST stuff?

REST is based on the following simple ideas:

- REST uses URIs to refer to and to access resources
- Uses HTTP methods to change the state of resources:

GET – retrieve details or a list of something

POST – create something on the server side

PUT / PATCH – update something on the server side

DELETE – delete something on the server side

And Who is this JSON guy?

XML	JSON

<person>
<first name>Johnny</firstname>
<last name>Userguy</lastname>
</person>
	{ "person":
	{
	"firstname": “Johnny",
	"lastname": “Userguy"
	}
	}

JSON (JavaScript Object Notation) is simply a way of passing data to a web page in a serialized way that is very easy to reconstitute into a javascript object.

JSON classes are built into every major javascript engine, so every browser has JSON encode/decode support.

```
{
  "name":"bigip-1-1",
  "protocol":"HTTP",
  "port": "80"
}
```

F5 iControl REST API (iCR)

- Starting with TMOS v11.5.0 we have introduced REST API for BIG-IP
  - A « kind of » remote TMSH through REST API and JSON formatted requests and responses
  - Acces to modules configuration and networking
  - As far as you have a TMSH command for it, you have a REST API entry for it
  - It is not only Configurations, it is also Statistics and more

Infrastructure

iControl REST

This release introduces a REST interface to iControl to remotely execute TMSH. iControl REST APIs are available for all BIG-IP product modules. TMSH versioning was added to provide script compatibility between versions of BIG-IP.

iControl REST in more details

- REST API calls are made of:
  - Authentication and RBACs
  - URI Format and REST entry point
  - HTTP Methods
  - JSON formatted content (request/response)

```
curl -sku Manager:Manager -X POST -H "Content-Type: application/json" -d
'{"username":"Manager", "password":"Manager", "loginProviderName":"tmos"}'
https://myBIGIP/mgmt/shared/authn/login | python -m json.tool
```

POST https://{{BIGIP}}/mgmt/shared/authn/login Params Send Save

form-data x-www-form-urlencoded raw binary JSON (application/json)

```
{
    "username":"Manager",
    "password":"Manager",
    "loginProviderName":"tmos"

}
```

Body Cookies (5) Headers (26) Tests Status: 200 OK Time: 62 ms

Pretty Raw Preview JSON Save Response


```
{
  "username": "Manager",
  "loginReference": {
    "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49f-82983026b41b/login"
  },
  "loginProviderName": "tmos",
  "token": {
    "token": "GH5FLVVDGBMVZMFNXMXBGX7XWA",
    "name": "GH5FLVVDGBMVZMFNXMXBGX7XWA",
    "userName": "Manager",
    "authProviderName": "tmos",
    "user": {
      "link": "https://localhost/mgmt/cm/system/authn/providers/tmos/1f44a60e-11a7-3c51-a49f-82983026b41b/users/f1fc
    },
    "groupReferences": [
      {
```

iControl REST in more details - AUTHENT

- Authentication and RBACs:
  - Our REST API implementation does not require to be ADMIN to use it
  - For Example, If you have a « Manager » Role, linked to a single partition, the user is given ONLY partition access/visibility
  - It requires multiple steps:
    1. Generate the authentication token through BIG-IP REST POST request
       - https://myBIGIP/mgmt/shared/authn/login, using your user credentials in the body and request basic auth (can be 3rd Party authent also)
    2. Retrieve the auth token generated in the response
    3. Use this token to retrieve information or set new configs

iControl REST in more details - AUTHENT

- Send authenticated request to get token:

```
curl -sku Manager:Manager -X POST -H "Content-Type: application/json" -d '{"username":"Manager",
"password":"Manager", "loginProviderName":"tmos"}' https://192.168.0.94/mgmt/shared/authn/login | python -m
json.tool | grep token
```

- From the response, with the token for REST calls:

```
 "token": {
        "kind": "shared:authz:tokens:authtokenitemstate",
        "selfLink": "https://localhost/mgmt/shared/authz/tokens/MZPCXQXKC6FPR26OTA2PVQHLMI",
        "token": "MZPCXQXKC6FPR26OTA2PVQHLMI",
```

- Send the GET request with the Token

```
curl -sk -X GET -H "X-F5-Auth-Token: MZPCXQXKC6FPR26OTA2PVQHLMI"
https://192.168.0.94/mgmt/tm/ltm/pool | python -m json.tool
```

iControl REST in more details - TRANSACTION

- What happens when you need to create multiple objects ?
- And if those objects are linked together ?
  - If one fails, most of the other SHOULD fail
  - If one fail, how can i revert the others ?

- In our Implementation, we support: « TRANSACTION »
- If the transaction is successful, all the REST operations were successful
- If not, all the REST commands are rolled back

iControl REST in more details - TRANSACTION

- Creating a transaction:

```
curl -sku admin:admin -X POST -H "Content-Type: application/json" -d '{}'
https://192.168.0.94/mgmt/tm/transaction | python -m json.tool

{
    "asyncExecution": false,
    "executionTime": 0,
    "executionTimeout": 300,
    "failureReason": "",
    "kind": "tm:transactionstate",
    "selfLink":
"https://localhost/mgmt/tm/transaction/1462779169531739?ver=12.0.0"
,
    "state": "STARTED",
    "timeoutSeconds": 120,
    "transId": 1462779169531739,
    "validateOnly": false
}
```

iControl REST in more details - TRANSACTION

- Getting a transaction status:

```
curl -sku admin:admin -X GET -H
"Content-Type: application/json" -d '{}'
https://192.168.0.94/mgmt/tm/transaction/1
462779377155939 | python -m json.tool
```


```
{
    "asyncExecution": false,
    "executionTime": 0,
    "executionTimeout": 300,
    "failureReason": "",
    "kind": "tm:transactionstate",
    "selfLink":
"https://localhost/mgmt/tm/transaction/1462779377155939?ver=12.0.0"
,
    "state": "STARTED",
    "timeoutSeconds": 120,
    "transId": 1462779377155939,
    "validateOnly": false
}
```

iControl REST in more details - TRANSACTION

- Adding commands to a transaction:

```
curl -sku admin:admin -X POST -H
"Content-Type: application/json" -H "X-F5-
REST-Coordination-Id:
1462779876798971" -d
'{"name":"pool_test","members": [
{"name":"5.6.7.8:80","description":"test
pool"} ] }'
https://192.168.0.94/mgmt/tm/ltm/pool |
python -m json.tool
```


```
{
    "body": {
        "members": [
            {
                "description": "test pool",
                "name": "5.6.7.8:80"
            }
        ],
        "name": "pool_test"
    },
    "commandId": 1,
    "evalOrder": 1,
    "kind": "tm:transaction:commandsstate",
    "method": "POST",
    "selfLink":
"https://localhost/mgmt/tm/transaction/1462779876798971/commands/
1?ver=12.0.0",
    "uri": "https://localhost/mgmt/tm/ltm/pool"
}
```

iControl REST in more details - TRANSACTION

- Checking a transaction content:

```
curl -sku admin:admin -X GET -H "Content-Type: application/json" -d '{}' https://192.168.0.94/mgmt/tm/transaction/1462779876798971/commands | python -m json.tool
```


```
{
    "items": [
        {
            "body": {
                "members": [
                    {
                        "description": "test pool",
                        "name": "5.6.7.8:80"
                    }
                ],
                "name": "pool_test"
            },
            "commandId": 1,
            "evalOrder": 1,
            "kind": "tm:transaction:commandsstate",
            "method": "POST",
            "selfLink":
"https://localhost/mgmt/tm/transaction/1462779876798971/commands/1?ver=12.0.0",
            "uri": "https://localhost/mgmt/tm/ltm/pool"
        } ],
    "kind": "tm:transaction:commandscollectionstate",
    "selfLink":
```

iControl REST in more details - TRANSACTION

- Commit of a transaction

```
curl -sku admin:admin -X PATCH -H
"Content-Type: application/json" -d '{
"state":"VALIDATING" }'
https://192.168.0.94/mgmt/tm/transaction/146
2779876798971 | python -m json.tool
```

SUCCESS

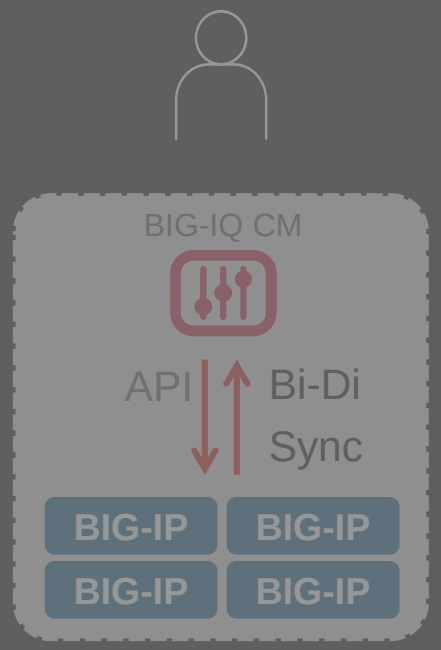

```
{
    "asyncExecution": false,
    "executionTime": 0,
    "executionTimeout": 300,
    "failureReason": "",
    "kind": "tm:transactionstate",
    "selfLink":
"https://localhost/mgmt/tm/transaction/1462781214151504?ver=12.0.
0",
    "state": "COMPLETED",
    "timeoutSeconds": 120,
    "transId": 1462779876798971 ,
    "validateOnly": false
}
```


FAIL

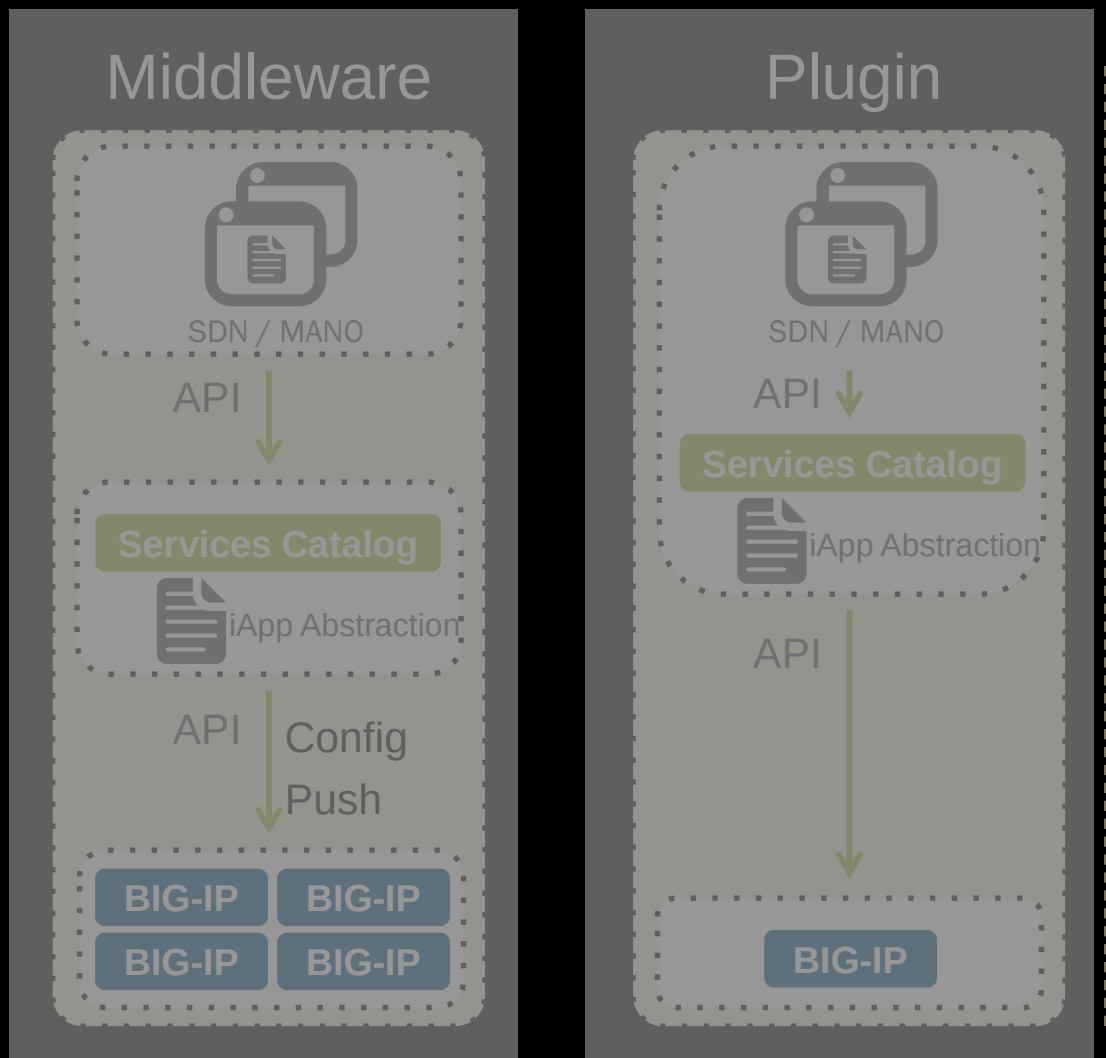

```
{
    "code": 409,
    "errorStack": [],
    "message": "transaction failed:01020066:3: The requested Pool
(/Common/pool_test) already exists in partition Common."
}
```

Different Orchestration Instantiations

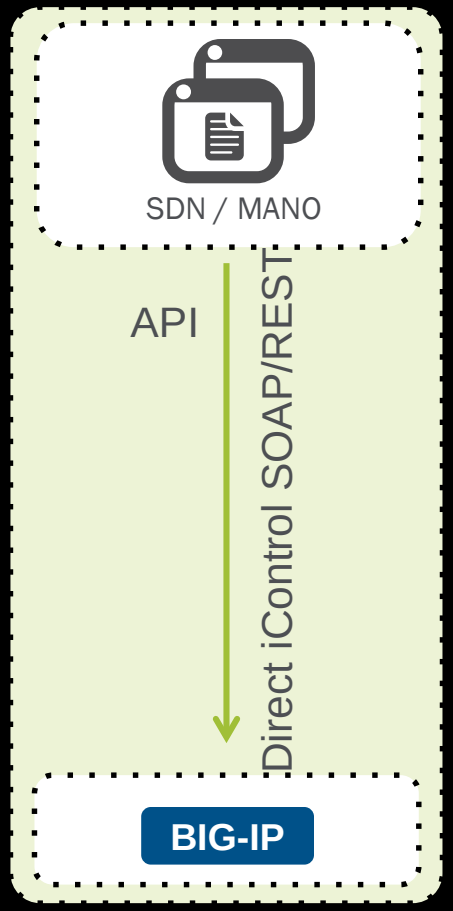
Central Mgmt



Orchestration Options



Classic



From REST API object calls to application deployment

From object creation to Application deployments

What we have seen ?

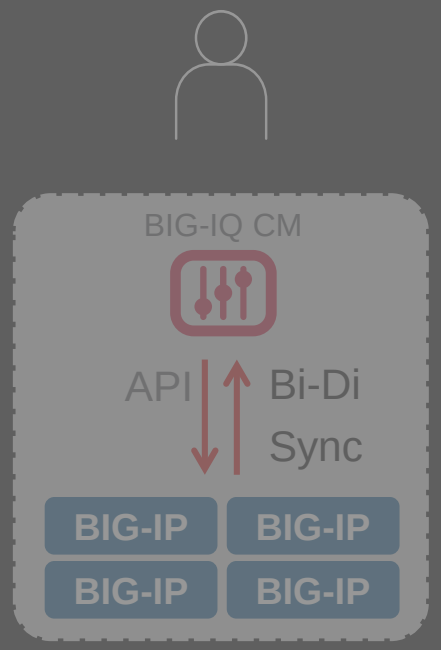
- ✧ REST API helps Create/Read/Update/Delete objects
- ✧ REST TRANSACTION help create an F5 config workflow

What is missing ?

- ✧ A user friendly interface for operation
- ✧ A way to simply modify objects (add or remove a pool member) with no complex CLI scripts

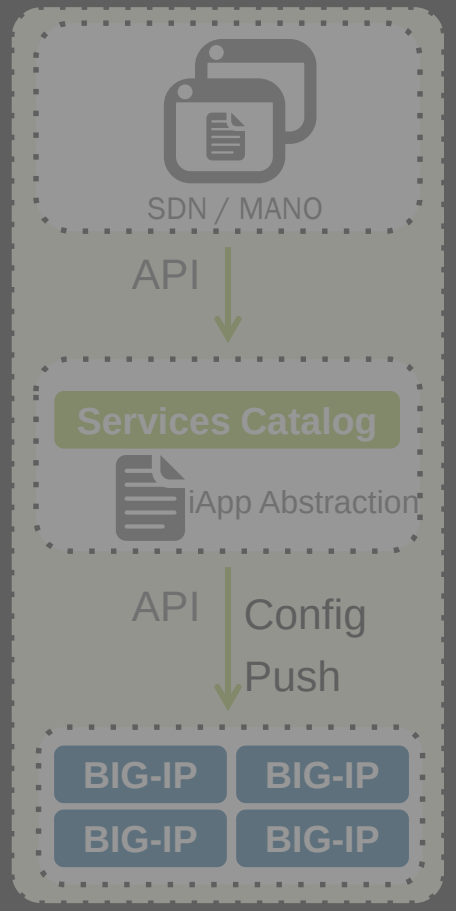
Different Orchestration Instantiations

Central Mgmt

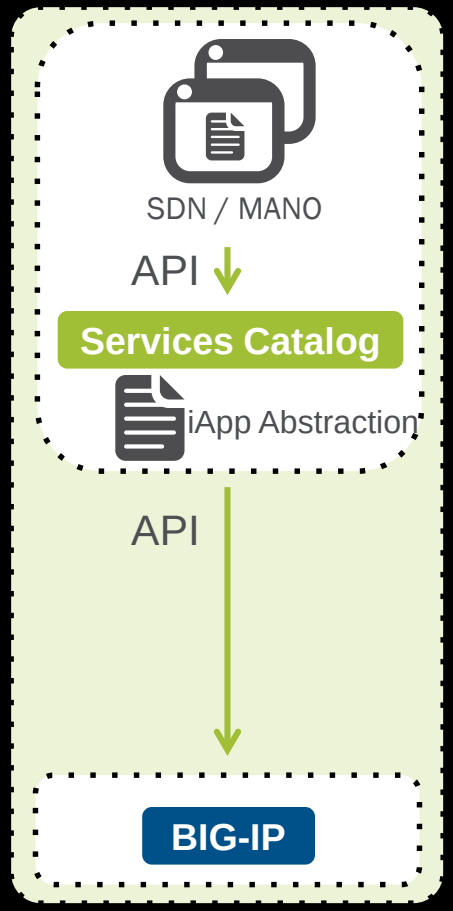


Orchestration Options

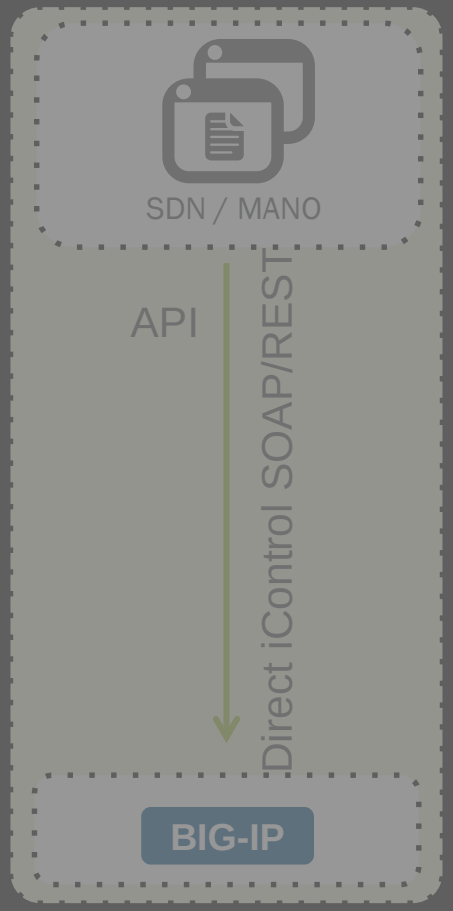
Middleware



Plugin



Classic



iApps provide different values depending on Application and Organization.

An App Lifecycle Tool

Unlike other template/wizard strategies, iApps are fully re-entrant, can manage the full lifecycle of the application.

A Single View App

Manage all application components in one place.

App Orchestration

Standardize your unique application deployments using iApps, iControl and BIG-IQ.

An Easy Button

Use F5-developed iApps to rapidly deploy popular applications with verified and supported configurations.

Standards Enforcement

iApps with strict updates, enforce standards, reducing training and operational risk.

iApp Templates under the hood

A formatted text file/script (.tmpl file) with three sections:

Implementation

Builds the config.

Written in TMSH and TCL.

Presentation

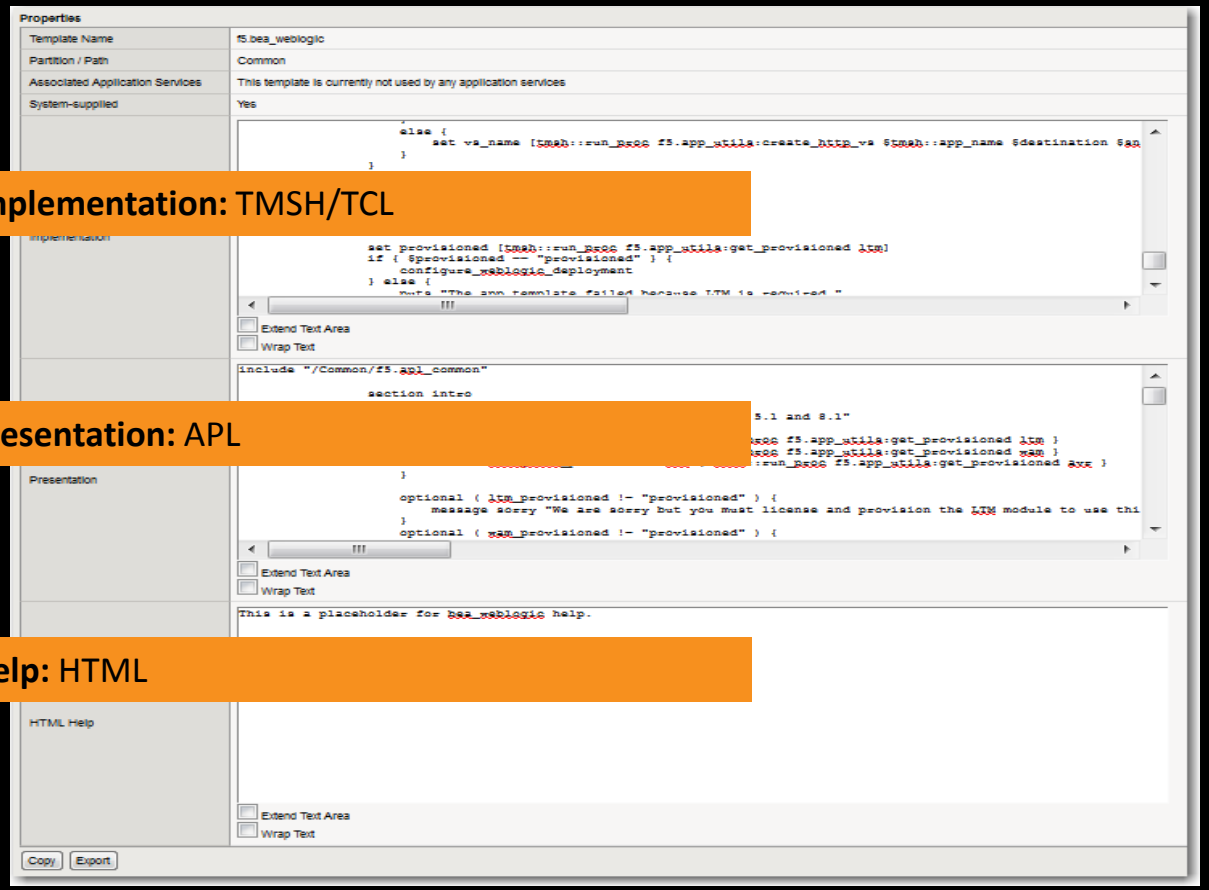
Defines the iApp wizard.

Written in APL.

Help

Documents the iApp.

Good, ol' HTML.

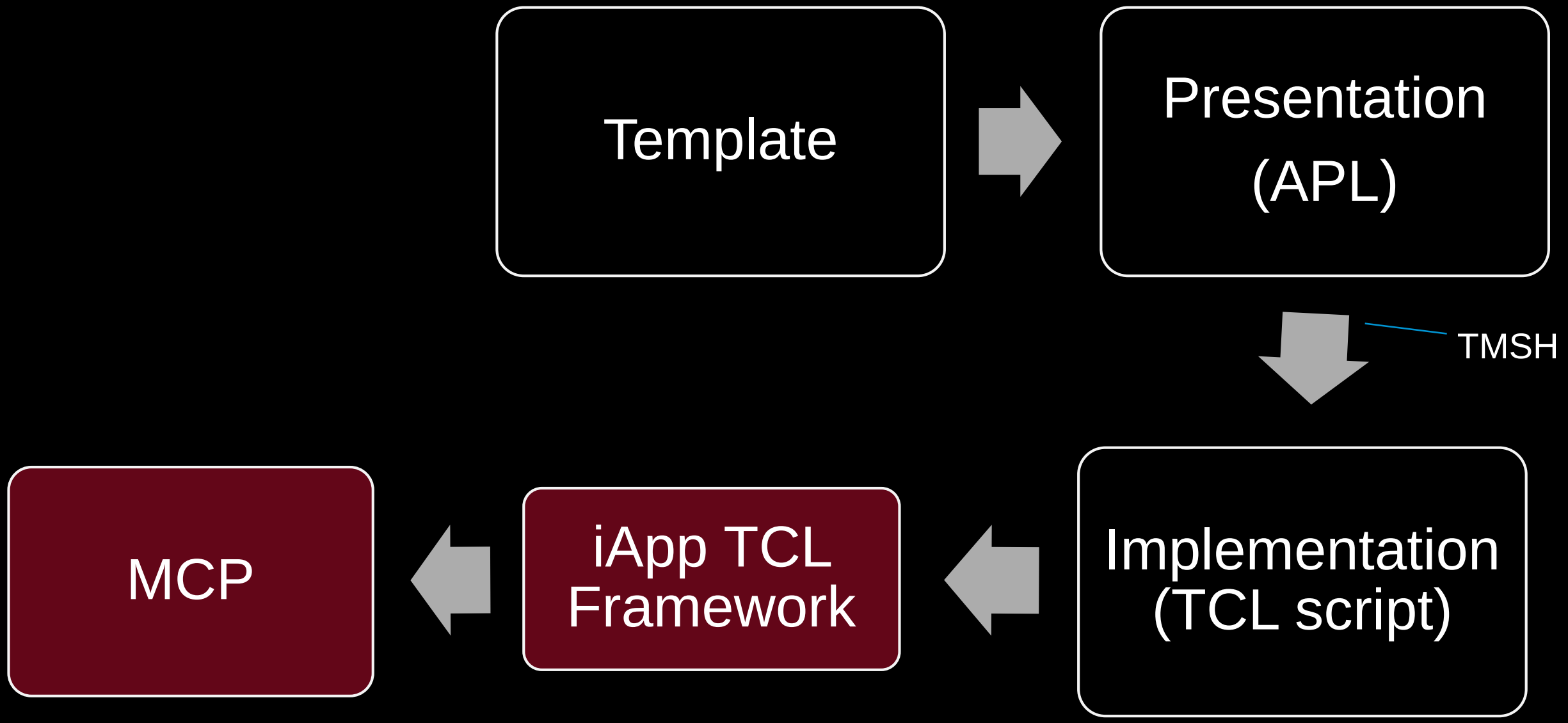


Controlling a portion of BIG-IP configuration

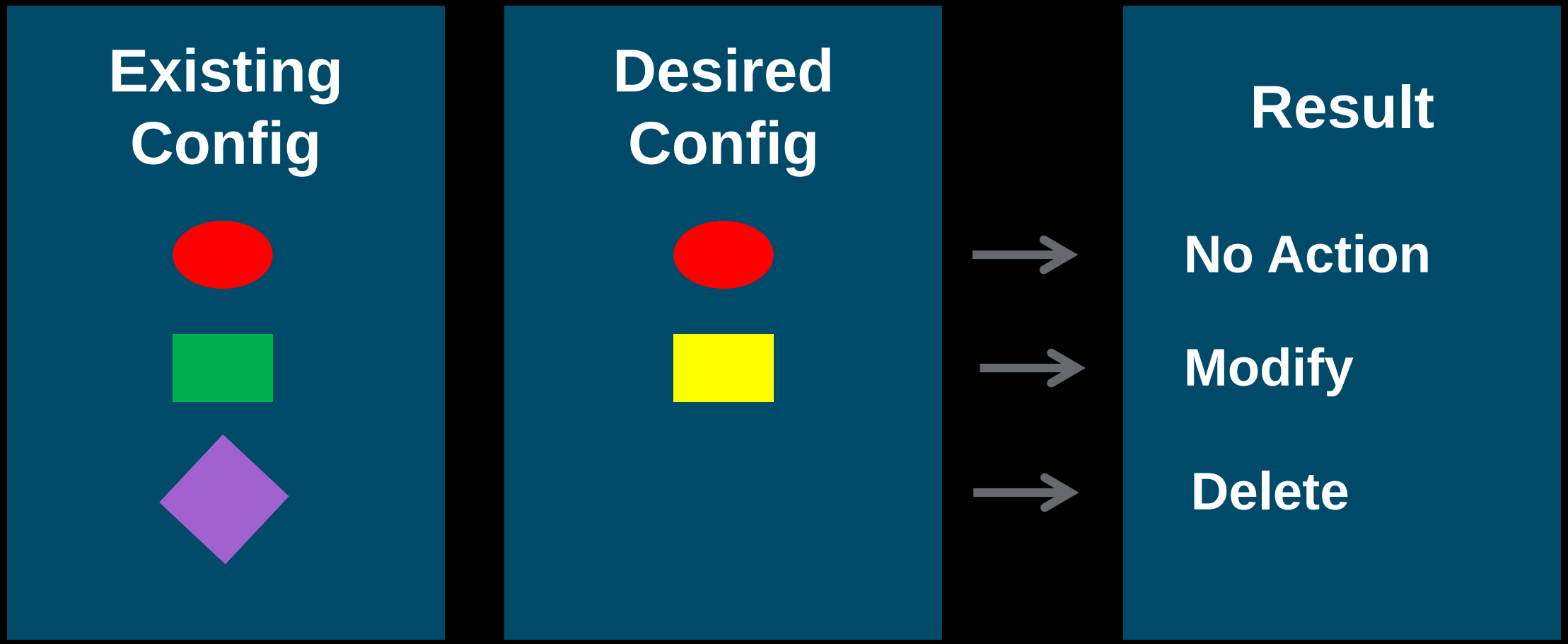
SaaS with SAML iApp

saas_v1		Application Service
saas_v1_encode_ObjectGUID_irule		iRule
saas_v1_vs	Unknown	Virtual Server
10.10.20.50		Virtual Address
saas_v1_http		Profile
saas_v1_client-ssl		Profile
wild_horizon.key		Certificate Key File
wild_horizon.crt		Certificate File
wild_horizon.crt.0		certificate_summary
wild_horizon.crt.1		certificate_summary
wild_horizon		clientssl_certkeychain
wild_horizon.crt		Certificate File
wild_horizon.crt.0		certificate_summary
wild_horizon.crt.1		certificate_summary
wild_horizon.key		Certificate Key File
saas_v1_wan-optimized-tcp		Profile
websso		Virtual Server Profile
rba		Virtual Server Profile
ppp		Profile

How do iApps work?



Mark and Sweep (re-configure iApp)



What REST and iApps can do for me ?

- iApp is defining a full (and sometimes complex) F5 configuration
- REST can be used to call an iApp with its relevant parameters
  - Multiple REST API calls for object creation can then be replaced by a single REST call (targetting the iApp)
  - Just the iApp entry fields can be used in the iApp REST call

REST and iApps: better together

Get the list of iApp templates:

```
curl -sku admin:admin -X GET -H "Content-Type: application/json" https://192.168.0.94/mgmt/tm/sys/application/template | python -m json.tool
{
    "items": [
 {
            "actionsReference": {
                "isSubcollection": true,
                "link": "https://localhost/mgmt/tm/sys/application/template/~Common~PIPO_HTTP/actions?ver=12.0.0"
            },
            "fullPath": "/Common/PIPO_HTTP",
            "generation": 24,
            "ignoreVerification": "false",
            "kind": "tm:sys:application:template:templatestate",
            "name": "PIPO_HTTP",
            "partition": "Common",
            "selfLink": "https://localhost/mgmt/tm/sys/application/template/~Common~PIPO_HTTP?ver=12.0.0",
            "totalSigningStatus": "not-all-signed",
            "verificationStatus": "none"
        },
```

REST and iApps: better together

Deploying a brand new WEB app:

```
curl -sku admin:admin -X POST -H "Content-Type: application/json" -d '{"kind":
"tm:sys:application:service:servicestate","name": "TEST2","template":
"/Common/PIPO_HTTP","strictUpdates": "enabled","partition": "/Common","tables": [{"columnNames":
["Pool_Member_IP","Pool_Member_Port"],"name": "Pool_definition__Pool_Members","rows": [{"row":
["5.6.7.8","80"]},{"row": ["7.8.9.10","443"]}]}],"variables": [{"encrypted": "no","name":
"Pool_definition__Pool_Name","value": "PoolName"},{"encrypted": "no","name":
"Pool_definition__existing_pool","value": "No"},{"encrypted": "no","name":
"Pool_definition__monitor_name","value": "/Common/http"},{"encrypted": "no","name":
"Virtual_definition__VS_IP","value": "5.4.4.4"},{"encrypted": "no","name":
"Virtual_definition__VS_Name","value": "VSName"},{"encrypted": "no","name":
"Virtual_definition__VS_Port","value": "8080"}]}' https://192.168.0.94/mgmt/tm/sys/application/service |
python -m json.tool
```

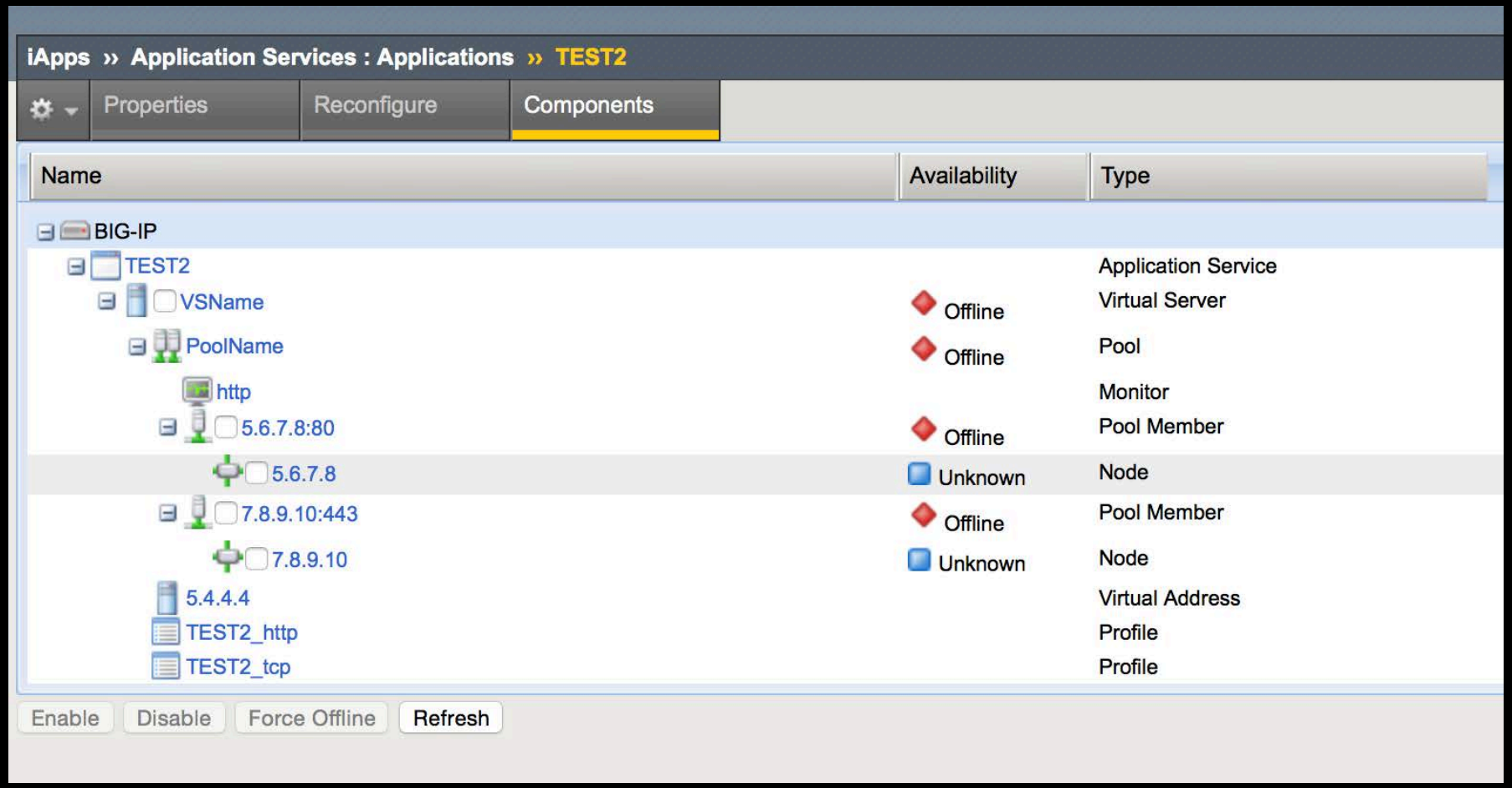
REST and iApps: better together

Deploying a brand new WEB app (response):


```
{
    "deviceGroup": "none",
    "fullPath": "/Common/TEST2.app/TEST2",
    "generation": 317,
    "inheritedDevicegroup": "true",
    "inheritedTrafficGroup": "true",
    "kind": "tm:sys:application:service:servicestate",
    "name": "TEST2",
    "partition": "Common",
    "selfLink": "https://localhost/mgmt/tm/sys/application/service/~Common~TEST2.app~TEST2?ver=12.0.0",
    "strictUpdates": "enabled",
    "subPath": "TEST2.app",
    "tables": [
        {
            "columnNames": [
                "Pool_Member_IP",
                "Pool_Member_Port"
            ],
            "name": "Pool_definition__Pool_Members",
```

REST and iApps: better together

What has been deployed:



REST and iApps: better together

- Need to program or automate this REST/iApp deployment ?
  - Use F5 provided SDKs

```
from f5.bigip import BigIP

# Connect to the BigIP
bigip = BigIP("192.168.0.94", "admin", "admin")

# Get a list of all pools on the BigIP and print their name and their
# members' name
pools = bigip.ltm.pools.get_collection()
for pool in pools:
    print ("["+pool.name+"]")
    for member in pool.members_s.get_collection():
        print ("-> "+member.name)
    print ("\r")
```

```
[HTTP_BAsic_pool]
-> 4.5.6.7:80

[Pool_TEST]
-> GoogleWeb1:80

[PoolName]
-> 5.6.7.8:80
-> 7.8.9.10:443

[PoolName]
-> 2.3.4.2:80
-> 5.6.7.8:80
-> 7.8.9.10:443

[PoolName]
-> 4.5.6.7:80
```

REST and iApps: better together

Docs » F5 Python SDK Documentation View page source

F5 Python SDK Documentation

build passing docs latest

Slack 4/36

Introduction

This project implements an object model based SDK for the F5 Networks® BIG-IP® iControl® REST interface. Users of this library can create, edit, update, and delete configuration objects on a BIG-IP®. For more information on the basic principals that the SDK uses, see the *User Guide*.

Quick Start

Installation

```
$> pip install f5-sdk
```

Note

If you are using a pre-release version you must use the `--pre` option with the pip command.

Basic Example

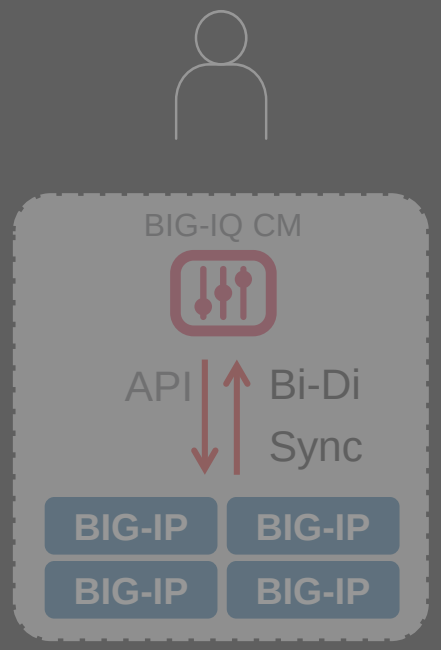
https://media.readthedocs.org/pdf/f5-sdk/latest/f5-sdk.pdf

REST and iApps: Better together

- How to make the « iApp » parameters/call, more « integrated » ?
- How to identify the parameters that need to be pushed for a specific App ? Tenant ?
- How to make this more « tenant » or App Owner compliant ? (no more access to CURL or scripts)?

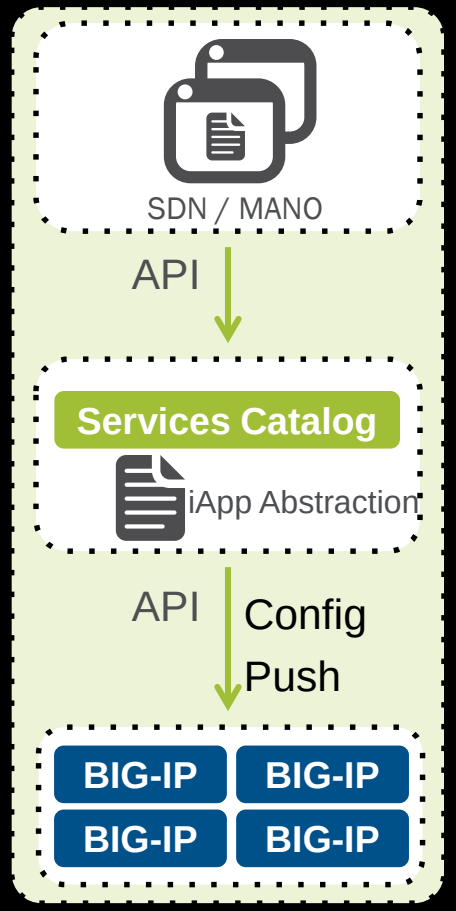
Different Orchestration Instantiations

Central Mgmt



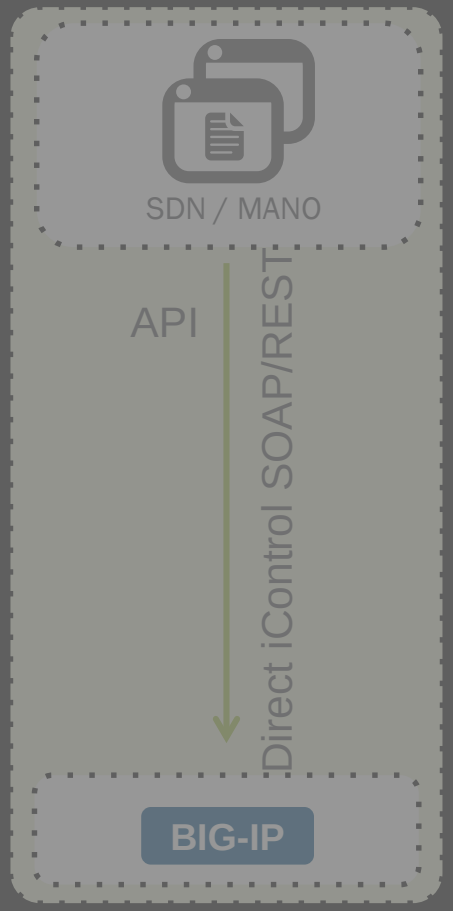
Orchestration Options

Middleware

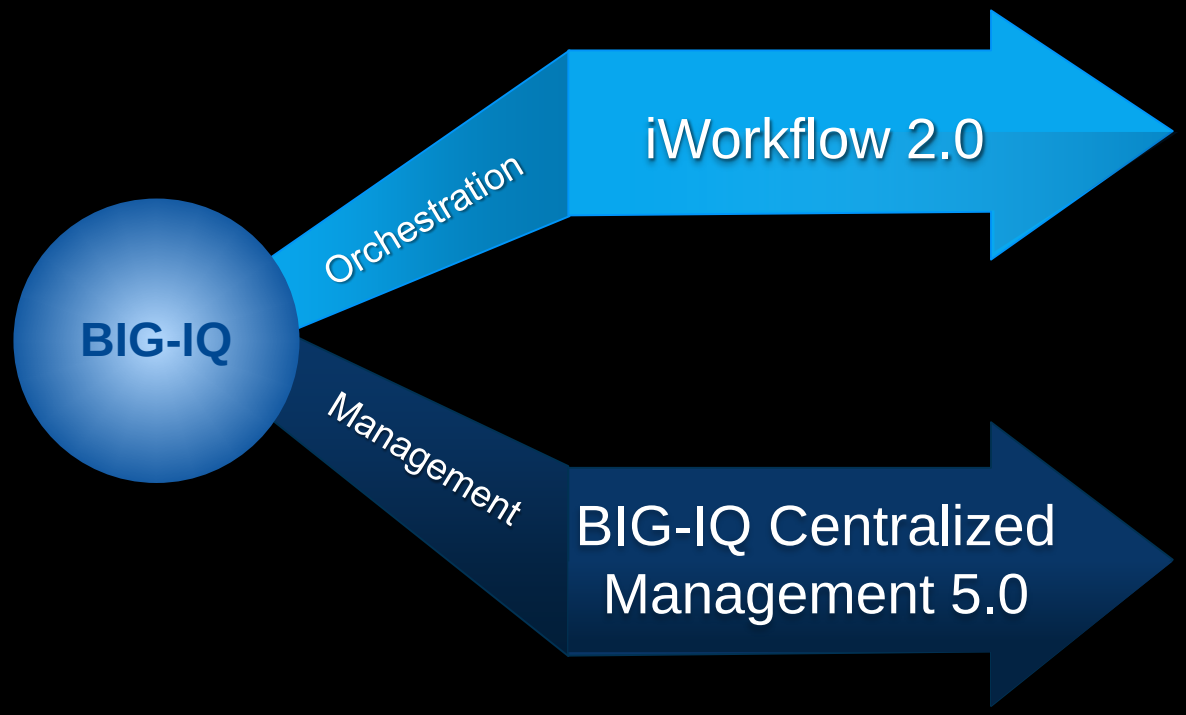


Plugin

Classic



F5 Management and Orchestration Strategy



iApp Catalogue

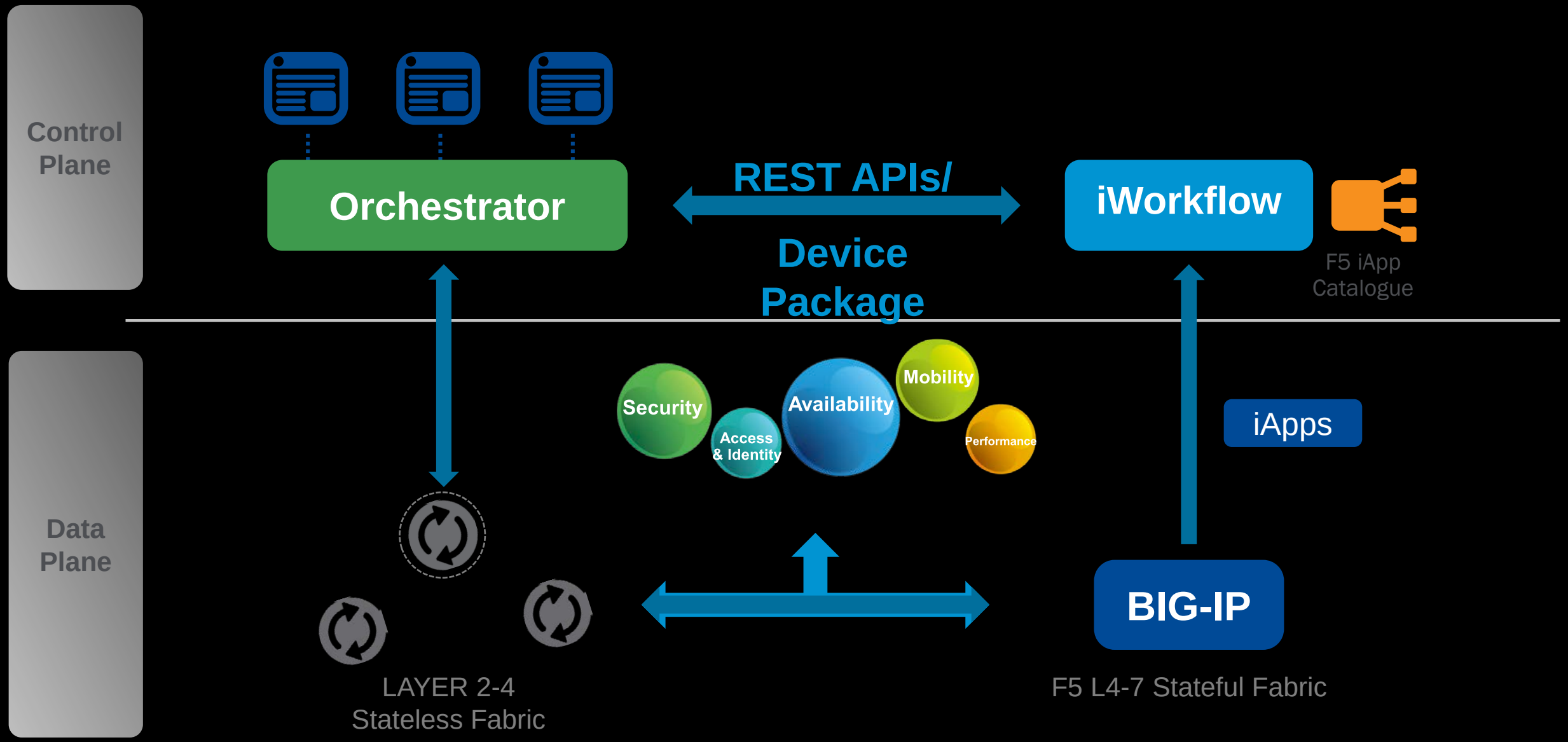
Rest Proxy

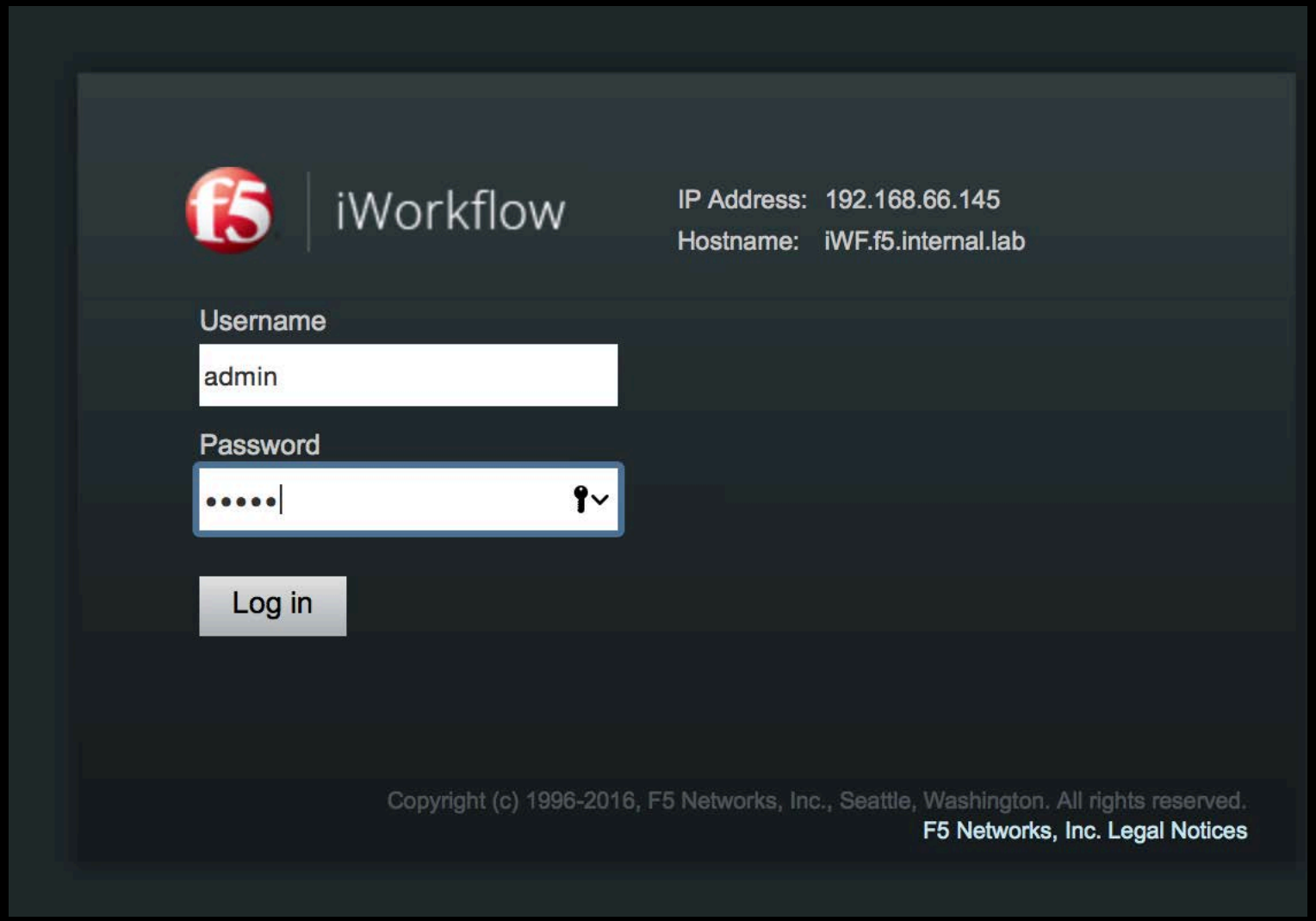
Enables programmatic management of many BIG-IPs

Single pane of glass BIG-IP management

Supports LTM, APM, AFM, ASM. Secure Web Gateway & Fraud Prevention

iWorkflow 2.0




iWorkflow
IP Address: 192.168.66.145
Hostname: iWF.f5.internal.lab
Username
admin
Password
Log in
Copyright (c) 1996-2016, F5 Networks, Inc., Seattle, Washington. All rights reserved.
F5 Networks, Inc. Legal Notices

Standalone

Hostname: iWF.f5.internal.lab | IP Address: 192.168

iWorkflow Cloud » Overview

Apply

Servers

2 items total

3.4.5.10:80
GENERATED | myAppv4

5.6.7.10:80
GENERATED | myAppv4

Connectors +

2 items total

ACI_Connector
Cisco APIC

myNewConnector
Local Cloud

New Connector

Basic Properties

Name

Description

Cloud Provider

- ✓ Select...
- Cisco APIC
- Local Cloud
- VMware NSX
- vCMP

Connectors +

2 items total

ACI_Connector
Cisco APIC

myNewConnector
Local Cloud

myNewConnector

Delete Save Cancel

Basic Properties

Name	myNewConnector
Description	
Cloud Provider	Local Cloud
Devices	192.168.66.123 - Standalone

Device & Server Provisioning

Timezone	Select...
NTP Server(s)	192.168.11.168
DNS Server(s)	192.168.255.1
DNS Suffix(s)	localdomain

Networks

	Subnet CIDR	Name	Gateway
Management Network			

Tenants +

1 item total

PIPOMOLO

Tenant Properties

Save Delete Cancel

Tenant Properties

Name	PIPOMOLO
Description	

Access

Available Connectors	myNewConnector

Contact Info

Address	
Phone	
Email	

Users +

2 items total

Admin User
admin - Local

pipomolo
pipomolo - Local

pipomolo

Save Delete Cancel

User Properties

Username	pipomolo
Full Name	pipomolo
Auth Provider	Local
Password	••••••
Confirm Password	••••••
User Groups	
User Roles	PIPOMOLO (Cloud Tenant)

Catalog +

0 items total

New Template

Tenant Preview | Save | Cancel

Input Parameters

- Accept Defaults
- Common Options
- All Options

Cloud Connector: myNewConnector

Application Type: PIPO_HTTP

Application Tier Information

Please match your application tier settings to the application properties below

Name: myAppTier

Pool: Pool_definition__P

SSL Cert: Select...

SSL Key: Select...

- Select...
- Pool_definition__Pool_Name
- Pool_definition__existing_pool
- Pool_definition__monitor_name
- Pool_definition__pools
- ✓ Virtual_definition__VS_IP
- Virtual_definition__VS_Name
- Virtual_definition__VS_Port
- app_stats

Sections

Expand All

Pool Selection 6/6

Catalog +

0 items total

New Template

Tenant Preview | Save | Cancel

Pool_definition__p...

Pool_definition__Pool...

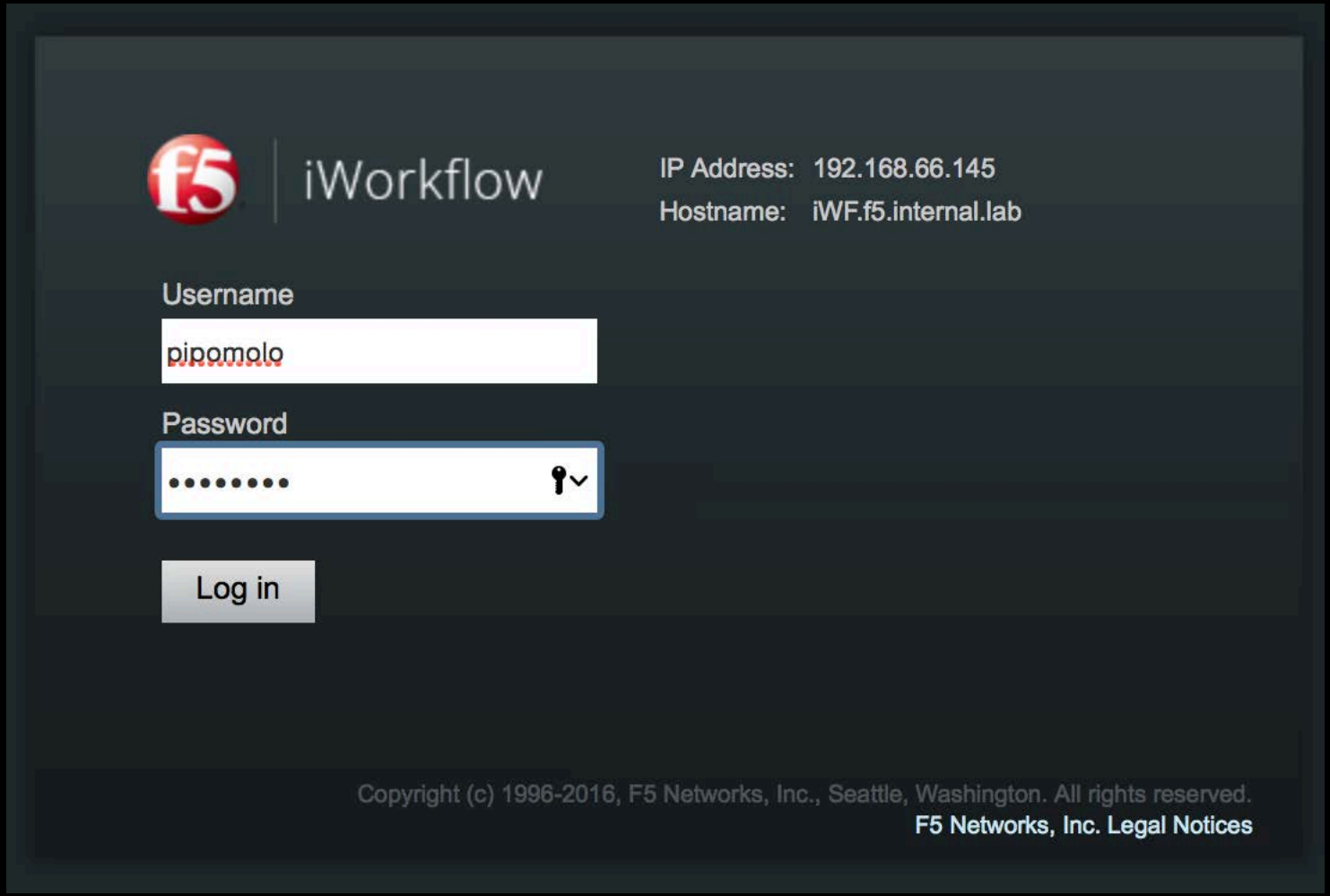
Enter a list of pool members or select an existing pool for the existing partition

Pool_Member_IP	Pool Member IP	☑
Pool_Member_Port	Pool Member Port	☑

Virtual Server Definition 3/3

Name	Description	Default Value	Tenant Editable
Virtual_definition_...	Virtual Server IP :		☑
Virtual_definition_...	Virtual Server Name:		☑
Virtual_definition_...	Virtual Server Port :		☑

General 1/1


iWorkflow
IP Address: 192.168.66.145
Hostname: iWF.f5.internal.lab
Username
pipomolo
Password
Log in
Copyright (c) 1996-2016, F5 Networks, Inc., Seattle, Washington. All rights reserved.
F5 Networks, Inc. Legal Notices

Applications +

0 items total

New Application

Save Cancel

General Properties

Name

Application Type

✓ Select...
myApp1

Applications +

0 items total

New Application

Save Cancel

General Properties

Name	myApp
Application Type	myApp1
Cloud Connector	myNewConnector

Customize Application Template

Virtual Server IP :	4.5.4.5	
Virtual Server Port :	80	
Enter a list of pool members or select an existing pool for the existing partition	Pool Member Ip	Pool Member Port
	3.3.4.4	80
	5.5.3.3	443
Virtual Server Name:	MYVsNAme	

Applications +

1 item total

myApp
PIPOMOLO

Servers

2 items total

3.3.4.4:80
GENERATED | myApp

5.5.3.3:443
GENERATED | myApp

Connectors

1 item total

myNewConnector
Local Cloud

Virtual Servers

1 item total

4.5.4.5
myApp

SOLUTIONS FOR AN APPLICATION WORLD