Table of Contents

[Lab 1 (AS3 – BIG-IQ) 1](#_Toc7389534)

[Lab 2 (Facts and AS3 with ansible – BIG-IP) 2](#_Toc7389535)

[Lab 3 (GIT) 3](#_Toc7389536)

[Lab 4 (POSTMAN) 4](#_Toc7389537)

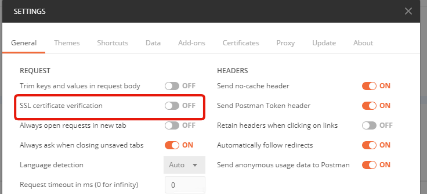
[Lab 5 (Validating declarations) 4](#_Toc7389538)

## Lab 1 (AS3 – BIG-IQ)

Goal

* Understand AS3 schema
* Create AS3 applications
* Read, update and delete AS3 applications
* Use the schema to create any type of application

Study the schema reference - [ <https://clouddocs.f5.com/products/extensions/f5-appsvcs-extension/latest/refguide/schema-reference.html> ] – (take some time going over it). This will be your guide when creating a new declaration.

* RDP to your windows machine in UDF
* Open postman (you can login to your account or not login at all), make sure to turn ssl verification off.
* 
* Import the provided collection and environment (desktop 🡪 lab files)

**[NOTE – postman should already be setup]**

* Use environment variables to create the provided apps
  + <https://hive.f5.com/docs/DOC-50438> (instructions on using the collection)
* Create the following applications using the collection provided (IP ranges for VS – 10.1.10.201 – 210)
  + HTTP
  + HTTP with LTM policy
  + HTTPs Offload
  + HTTPs Bridging
  + TCP
  + UDP

(observe the response, use the ID in the response to check the status of the deployment task)

* Read the above applications and observe the output using the collection provided
  + List all apps
  + List one app
  + If you were designing a page which supports CRUD, think optimization while listing multiple apps
* Study the jinja2 templating used (in postman)
* Create the following applications (not provided) [ use the schema]
  + HTTP with SNAT pool
  + Based on SSL SNI match conditions, steer traffic to separate pools
  + A declaration using the metadata – how can you leverage metadata?
  + Compare declarations created using HTTP template and generic template
  + Can you disable a pool member when using HTTP template Vs Generic template?
* Study and play with the age parameter (<https://clouddocs.f5.com/products/extensions/f5-appsvcs-extension/latest/refguide/as3-api.html?highlight=history#method-get>)
* Study the “AS3 will all properties” page (have it as a ref) - <https://clouddocs.f5.com/products/extensions/f5-appsvcs-extension/latest/declarations/all-properties.html>

## Lab 2 (Facts and AS3 with ansible – BIG-IP)

Goal – Use ansible for BIG-IP fact gathering and creating applications using AS3

1. bigip\_device\_facts – Collect facts from F5 BIG-IP devices

<https://docs.ansible.com/ansible/latest/modules/bigip_device_facts_module.html#bigip-device-facts-module>

Device facts outputs data about existing BIG-IP devices in JSON format. For example, it can provide data about virtual servers, pools, ssl-certs, profiles etc. This data can be leveraged to analyze the existing environment to understand the existing applications (specially in large environments). Also, since its JSON it can be programmatically leveraged.

Lab steps

* Go to the directory /home/ubuntu/ansible/techdayansible/lab-bigip-facts
* Run the playbook (ansible-playbook -i inventory/hosts playbooks/site.yaml)
* Watch the output (on the screen as well as in the host\_vars), what is it obtaining?
* Change the playbook to get facts for what you want (see docs above)

## Lab 3 (GIT)

Goal – Understand basic GIT concepts by experimenting with GIT bash and GIT GUI (you can use any GUI of your choice)

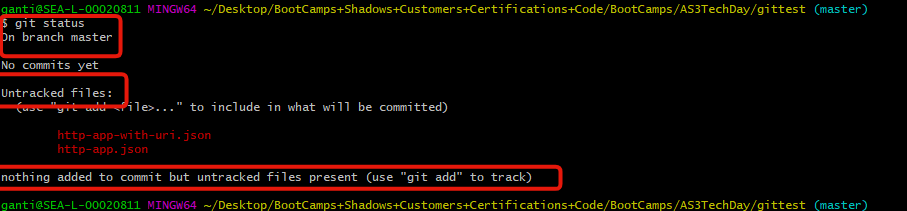
Basic git bash steps - <https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html>

Note – you can right click and open ‘git bash’ and ‘git gui’ in the ‘gittest’ folder.

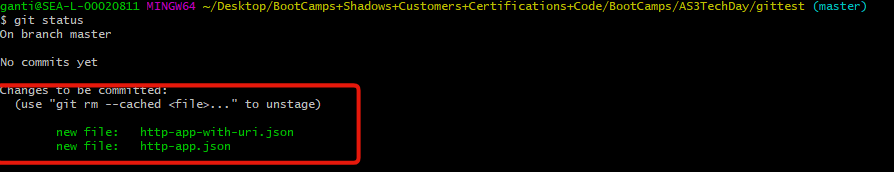
Lab steps

[basic lab] – refer to commands in the link above

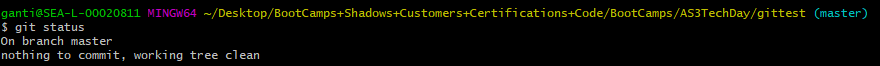
* Create a bitbucket/github account (if you do not have already)
* Add your public key to your account (~/.ssh/id\_rsa.pub). If it is not present, create it by using ssh-keygen
* Initialize a repo in the ‘gittest’ folder(desktop) using git bash. Notice what happens in the folder after you initialize the repository.
* Set name and email globally using git config. This will be used to just as an identifier when committing to the repository.
* Check status



* Add all files to the staging area
* Check status



* Commit with a message
* Check status



* Create a repo in bitbucket/github
* Add the repo created as a remote in your local repository
* Check the remote got added

https://i.gyazo.com/ce9a9873ba205ddebb46b9eae6ec1ac8.png

* Push to remote, verify the remote repository that the files got added.
* Update the http file (change any value you want)
* Open GIT GUI
* Check the un staged, staged and modified areas
* Stage the changed file, notice the change in the windows
* Commit with a new message (e.g. – modified IP address of VIP”)
* Push to remote, notice the commits section of the remote repository has the new commit now.



* Click on the hash value and notice your change
* Create a file named ‘.gitignore’ with the body ‘\*.txt’
* Create two files with random content, name them test.json and testtext.txt
* Add all files in tracking again, notice which files get added. What does adding ‘\*.txt’ to ‘.gitignore’ do? What other type of files can you think of that fit the similar use case ?

## Lab 4 (POSTMAN)

Goal – Get comfortable with postman

Lab steps

* Observe the collection used for the AS3 lab
* Examine the templating used
* Check out the pre\_request\_scripts tab, what is it doing?
* What do tests do? Can you try to write a sample?

## Lab 5 (Validating declarations)

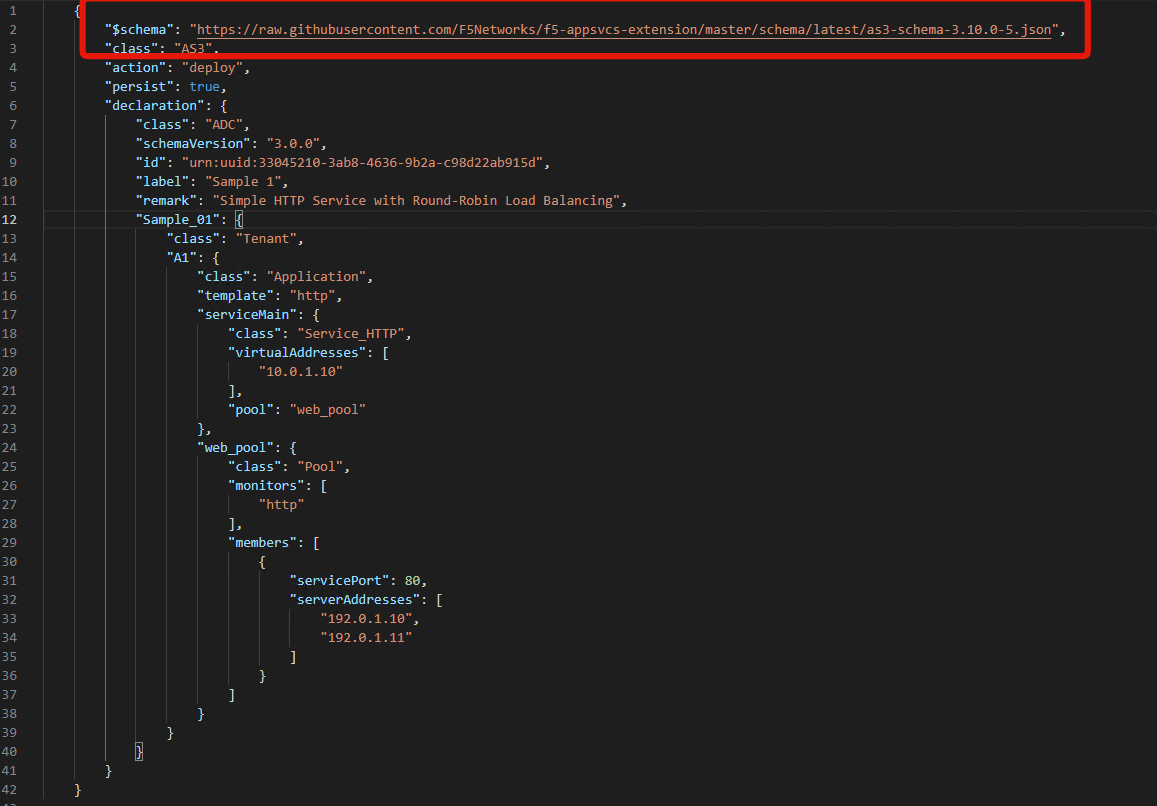
Goal – Explore validating declarations

1. VS code for validating declarations – This is used for manual validation of declarations. This comes with F5 official docs.

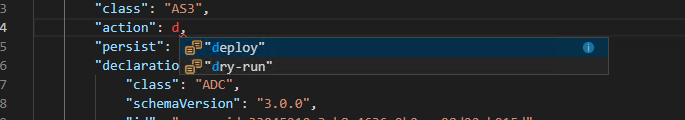
<https://clouddocs.f5.com/products/extensions/f5-appsvcs-extension/latest/userguide/validate.html>

Lab steps

* Study the link above
* Open VS code on the desktop
* Open the file VScode-validation.json (lab-files)
* Look at the $schema line up top (pointing to the latest schema)



* Notice that as you write the declaration, the editor will give you suggestions



* Notice that the editor will also do syntax highlighting if the declaration syntax is incorrect



1. Validation declarations programmatically – Most of the time, the declarations will be handled by a piece of code, and there should be a way of programmatically verifying if the declaration is valid or not. There are some 3rd party validators available for that. Example of one of them is - <https://github.com/Julian/jsonschema>

## Extra labs

If your familiarity level with the above labs is high, then you can try these.

1. [medium difficulty] CI/CD Demo lab on UDF
2. [high difficulty] Write a library in the language of your choice supporting CRUD operations for applications. The library would be a class which can be instantiated by passing the credentials of BIG-IQ (IP, user, pass). Some of the supported functions to think about are the following (think of what data should be input, and how the response should be returned).

* Listing all declarations
* Listing one declaration in detail
* Creating app
* Dry run apps
* Deleting apps