**NETWORK AUTOMATION**

**USING**

**PYTHON RESTAPI**

**A PROJECT REPORT**

***Submitted by***

**INDHU J (412718104007)**

**ANUPRIYA M (412718104001)**

***in the partial fulfilment for the award of the degree***

***Of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

**MAY, 2021**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“NETWORK AUTOMATION USING PYTHON RESTAPI ”** is the bonafide work of **“INDHUJ”(412718104007) and “ANUPRIYA M” (412718104001), and** who carried out the project work undersupervision.

|  |  |
| --- | --- |
| **SIGNATURE**  **Dr . S. SURENDRAN,M.E.., Ph.D.**  **HEAD OF THE DEPARTMENT ,**  **Professor,**  Department of CSE,  Tagore Engineering College,  Rathinamangalam.  Vandalur ,Chennai-600127 | **SIGNATURE**  **Mr. SUDHEER REDDY BANDI,M.E SUPERVISOR,**  **Assistant Professor,**  Department of CSE,  Tagore Engineering College,  Rathinamangalam  Vandalur ,Chennai-600127 |
|  |  |  |
|  |  |

**ACKNOWLEDGEMENT**

We take this opportunity to extend our sincere and hearty thanks to our chairperson for providing us an opportunity and faculties to purpose our course in this institution.

We express our gratefulness to **Dr.L.RAJA.,** The Principal, Tagore Engineering College for this constant encouragement and guidance.

We express our special thanks to **Dr.S.SURENDRAN .M.E., Ph.D,** Professor and Head, Department of Computer Science Engineering, Tagore Engineering College for the continuous support and guidance. We extend our cordial thanks to our internal guide for helping us to complete it, **MR. SUDHEER REDDY BANDI** and our project coordinators **Dr.S.SURENDRAN .M.E.,Ph.D,** Tagore Engineering College, for their role in completion of our project work.

We also thank all the faculty members, lab technicians and friends who have given us great

**J. INDHU (412718104007)**

**M.ANUPRIYA (412718104001)**

**ABSTRACT**

The Cisco Application Centric Infrastructure (ACI) provides powerful new ways to dynamically manage infrastructure in the modern world of IT automation and DevOps. Having the tools to change how infrastructure is built is one thing, but being able to effectively operate the infrastructure beyond the day zero build activities is crucial to long term effectiveness and efficiency. To effectively harness the power of ACI, organizations will need to understand how to incorporate ACI into their daily operations.

This book examines some of the common operational activities that IT teams use to provide continued infrastructure operations and gives the reader exposure to the tools, methodologies, and processes that can be employed to support day 1+ operations within an ACI-based fabric.

The Application Policy Infrastructure Controller (APIC) REST API is a programmatic interface that uses REST architecture.

The API accepts and returns HTTP (not enabled by default) or HTTPS messages that contain JavaScript Object Notation (JSON) or Extensible Markup Language (XML) documents. You can use any programming language to generate the messages and the JSON or XML documents that contain the API methods or Managed Object (MO) descriptions.

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO**

**ABSTRACT** iii

**LIST OF TABLE xvi**

**LIST OF FIGURES** xviii

**1. INTRODUCTION**

1.1 GENERAL **10**

**2.** **System analysis 11**

**Existing System**

**Proposed System**

**4. SYSTEM REQUIREMENTS 15**

3.1HARDWARE REQUIREMENTS

3.2 SOFTWARE REQUIREMENTS

**5. Python 16**

**6. Introduction to Python 17**

**7.** **Python Libraries 21**

**8. Introduction to Project 25**

**8. SYTEM DESIGN 30**  **MODULES DESCRIPTION 40**

**9. SYSTEM TESTING 50**

**10. CONCLUSION AND FUTURE ENHANCEMENT 55**

**APPENDIX 1**

**11 ABSTRACT**

**12 LIST OF TABLE xvi**

**13 LIST OF FIGURES xviii**

|  |  |  |
| --- | --- | --- |
|  | **LIST OF FIGURES** |  |
| **FIGURE.NO** | **TITLE** |  |
| 5.1 | System Diagram |  |
| 5.2 | System Design |  |
|  |  |  |
|  |  |  |

**LIST OF ABBREVIATIONS**

**ABBREVIATIONS                          ACRONYMS**

**APIC** Application policy Infrastructure controller

**REST** Representational  state transfer

**API** Application Programming Interface

**POST** Power on self test

**CHAPTER 1**

**GENERAL** To apply Rest API Techniques resulting in improving the

Network automation.

**SYNOPSIS**

The Application Policy Infrastructure Controller (APIC) REST API is a programmatic interface that uses REST architecture. The API accepts and returns HTTP (not enabled by default) or HTTPS messages that contain JavaScript Object Notation (JSON) or Extensible Markup Language (XML) documents. You can use any programming language to generate the messages and the JSON or XML documents that contain the API methods or Managed Object (MO) descriptions.The REST API is the interface into the management information tree (MIT) and allows manipulation of the object model state.

The same REST interface is used by the APIC CLI, GUI, and SDK, so that whenever information is displayed, it is read through the REST API, and when configuration changes are made, they are written through the REST API. The REST API also provides an interface through which other information can be retrieved, including statistics, faults, and audit events. It even provides a means of subscribing to push-based event notification, so that when a change occurs in the MIT, an event can be sent through a web socket.

Standard REST methods are supported on the API, which includes POST, GET, and DELETE operations through HTTP. The POST and DELETE methods are idempotent, meaning that there is no additional effect if they are called more than once with the same input parameters. The GET method is nullipotent, meaning that it can be called zero or more times without making any changes (or that it is a read-only operations

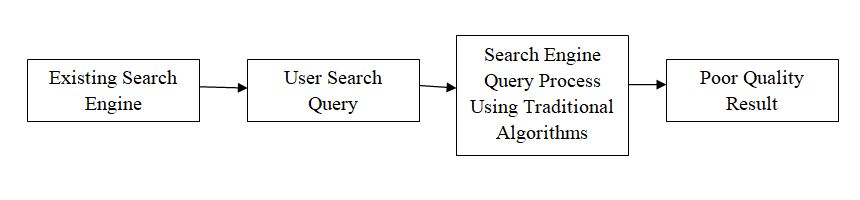
**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

To develop web based Search Engine application using python scripting language to provide an excellent searching platform on the internet for the users.

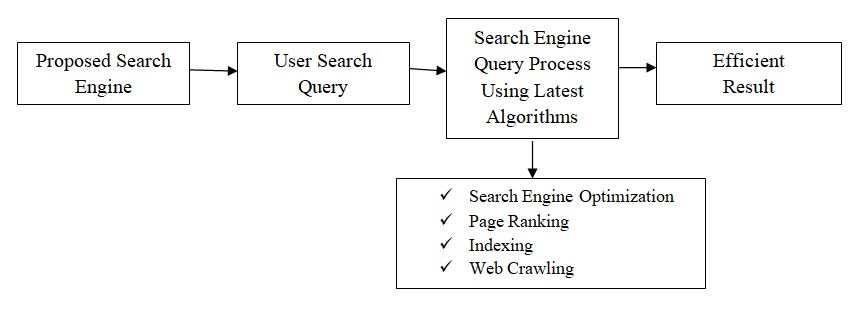
The traditional Search Engine was able to produce the results for the user’s requirements. But the question of efficient result is the biggest challenge for any search engine project. Few times only users happy with the search results given by search engines. Most of the time search engine failed to produce the required search result requested by the user. Most of the users not very much satisfied with the existing system.

*Figure: Existing System Architecture[](https://images.projectsgeek.com/2018/01/Search-Engine-using-Python-flow.jpg)*

**2.2 PROPOSED SYSTEM**

In order to solve the problems in the existing system, the proposed system uses the latest algorithms such as search engine optimization technique, page ranking, indexing and web crawling. This will provide the best search results to the users. The unique and distinct search result is displayed by the proposed search engine to the user’s query.

A number of web links are indexed, whenever a keyword is searched by the user using this Search Engine application. The analysis is performed for the data in the indexed links. This task is achieved by page rank algorithm. Finally, the web page containing the top match to the key word is showedto the user in the first link of the result page.



**OVERVIEW OF METHOD AND RESULTS**

There are a number of fundamental differences between the operations of traditional networking hardware and an Cisco Application Centric Infrastructure (ACI) fabric.

These differences serve to simplify the management greatly, reduce the number of touch points, and decouple the switching hardware from the desired configuration intent.

These changes include:

* Single point of management controller-based architecture
* Stateless hardware
* Desired state-driven eventual consistency model

The single point of management within the ACI architecture is known as the Application Policy Infrastructure Controller (APIC). This controller provides access to all configuration, management, monitoring, and health functions. Having a centralized controller with an application programming interface (API) means that all functions configured or accessed through the fabric can be uniformly approached through a graphical user interface (GUI), command line interface (CLI), and API, with no risk of inconsistency between the various data interfaces. This results in a clean and predictable transition between the interfaces. The underlying interface for all access methods is provided through a REST-based API, which modifies the contents of a synchronized database that is replicated across APICs in a cluster and provides an abstraction layer between all of the interfaces.

This controller-based architecture also makes possible a stateless configuration model that decouples the hardware from the configuration running on it.

This translates to an APIC cluster that manages individual fabric nodes of leaf and spine switches that derive their identity from what the controller defines as being the desired intent, and not from the serial number of the chassis, nor from a configuration file residing on the devices.

Each node receives a unique node identifier, which allows for the device to download the correct configuration attributes from the controller.

The device can also be substituted in a stateless fashion, meaning that hardware swaps can be faster, topology changes are less impactful, and network management is simplified.

The desired state model for configuration further complements these concepts of controller-based management and statelessness by taking advantage of a concept known as declarative control-based management, based on a concept known as the promise theory.

Declarative control dictates that each object is asked to achieve a desired state and makes a "promise" to reach this state without being told precisely how to do so. This stands in contrast with the traditional model of imperative control, where each managed element must be told precisely what to do, be told how to do it, and take into account the specific situational aspects that will impact its ability to get from its current state to the configured state.

A system based on declarative control is able to scale much more efficiently than an imperative-based system, since each entity within the domain is responsible for knowing its current state and the steps required to get to the desired state, dictated by the managing controller.

**CHAPTER 2**

**REQUIREMENT SPECIFICATIONS**

**2.1 INTRODUCTION**

ACME Inc. is a multi-national **corporation** that specializes in manufacturing, sales, and distribution of a diverse product portfolio, including rocket-powered roller skates, jet-propelled unicycles, and various explosive materials.

These product groups operate as separate business groups within the company, and have previously maintained separate infrastructure and applications.

They have largely focused on retail routes to market, but have recently decided to pursue a more direct-to-consumer business model due to intense pressure from new competitors who have dominated the online sales channels.

Traditionally, ACME business units have leveraged third party software companies and commercially available software to meet their IT demands, but would like to create a more intimate relationship with their consumers and be able to take feedback on the platform directly from those users, while incorporating an ongoing improvement cycle so they can react to changing market dynamics in a more nimble fashion.

Where they have used custom software in the past, they have leveraged a traditional infrastructure and software model that does not allow them to keep up with the changing requirements, and therefore ACME is looking for a new approach to both application and infrastructure life cycle management.

The application developers have been looking at new application development trends such as Continuous Delivery and Continuous Integration, and the new application platform is to be developed in this manner. To support this, the infrastructure components need to be capable of mapping to these new paradigms in a way that is not possible using traditional concepts.

One of the largest challenges ACME has historically faced is that operations and infrastructure has been an afterthought to product development. This has led to several situations where application deployments have meant long weekend hours for all of the teams, caused customer-impacting outages, and taken longer to accomplish than the business leaders would have liked. For this reason, ACME Inc. has decided to change by creating an environment where infrastructure artifacts are treated as part of the application, can be checked into version control, can be tested alongside the actual application, and can continually improve.

While ACME is intensely focused on delivering the new application platform in a timely manner, ACME is also interested in creating a foundation on which it can grow to deliver a common pool of infrastructure that is shared across all business groups and operated in a multi-tenant fashion to increase efficiency.

At an executive briefing, John Chambers, the CEO of Cisco Systems at the time, told ACME: "The world is changing. Every company is a technology company, and if you don't adapt, you'll get left behind."

As evidenced by the success of cloud platforms, such as Amazon Web Services (AWS) and OpenStack, consumption models of technology delivery have the ability to adapt technology more quickly to rapid business requirements changes. This is the type of consumption that ACME Inc.'s business owners need. Control of operations is what operations groups are focused on, but control can be a barrier to a pure consumption model. Unless companies make investments in technologies that allow for consumption of automated components, the only other way to scale is by breaking the human level component, and few people would really choose to work for that type of company.

After analyzing current offers from various technology vendors, ACME Inc. selected Cisco Application Centric Infrastructure (ACI). The ability to abstract all physical and virtual infrastructure configuration into a single configuration that is consistent across dev, test, and prod environments, as well as portable across the various data center locations currently maintained by ACME, is highly desirable. ACI has been built from the ground up to change the substructure used to build network devices and protocols.

Innovation at this level will provide more opportunities for expanding the tools with which users interact.

This is where the fulcrum will tilt in the favor of IT and infrastructure being more dynamic, thus allowing IT to operate and manage at the speed of business. However, with a change of this nature comes fear, uncertainty, and doubt.

This book will attempt to bring some level of comfort and familiarity with operations activities within an ACI fabric.

While ACME Inc. is a fictitious company, this is the true story of every company, and just important this is the story of the employees of those companies.

Workers in the IT industry need to adapt to keep up with the rapid change of the business.

However, this runs contrary to how most operations groups exist in the relationship between business and technology.

Most IT operations groups invest a lot of time in the tools needed to deliver services today and there is an organic resistance to re-investing.

**2.2 HARDWARE AND SOFTWARE SPECIFICATIONS**

**2.2 HARDWARE REQUIREMENTS**

* Hard Disk - 500GB and Above
* RAM - 8GB and Above
* Processor - I4 and Above

**2.2.2 SOFTWARE REQUIREMENTS**

* Operating System - Windows 7, 8, 10(64 bit)
* Software - Python 3.9.5
* Tools - Visual studio code (IDE)

**2.3 TECHNOLOGIES USED**

* Python
* Networking Automation

**2.3.1 INTRODUCTION TO PYTHON**

Python is a widely used general-purpose, high level programming language. It was

initially designed by Guido van Rossum in 1991 and developed by Python Software

Foundation.

It was mainly developed for emphasis on code readability, and its syntax allows programmers

To express concepts in fewer lines of code.

**It is used for**

* web development (server-side),
* software development,
* mathematics,
* System scripting.

**What can Python do?**

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.

**Why Python?**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

**Good to know**

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been backported to Python 2. But in general, they remain not quite compatible.
* Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official [End Of Life date of January 1, 2020](https://pythonclock.org/) has been established for Python 2, after which time it will no longer be maintained.
* Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe [Monty Python’s Flying Circus,](https://en.wikipedia.org/wiki/Monty_Python%27s_Flying_Circus) of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.
* It is possible to write Python in an Integrated Development Environment, such as Thonny, Visual Studio, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

**Python Syntax compared to other programming languages**

* Python was designed to for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**Python is Interpreted**

* Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer’s processor, before it can be run.

Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

* This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.
* One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.
* In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.
* For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and [object-oriented programming.](https://realpython.com/python3-object-oriented-programming/)
* Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

**Python Goals :**

* an easy and intuitive language just as powerful as those of the **major** competitors;
* open source, so anyone can contribute to its development;
* code that is as understandable as plain English
* suitable for everyday tasks, allowing for short development times.

**Python is a great choice for**:

* **Web and Internet development** (e.g., Django and Pyramid frameworks, Flask and Bottle micro-frameworks)
* **Scientific and numeric computing** (e.g., SciPy – a collection of packages for the purposes of mathematics, science, and engineering; Ipython – an interactive shell that features editing and recording of work sessions)
* **Education** (it’s a brilliant language for teaching programming!)
* **Desktop GUIs** (e.g., wxWidgets, Kivy, Qt)
* **Software Development** (build control, management, and testing – Scons, Buildbot, Apache Gump, Roundup, Trac)
* **Business applications** (ERP and e-commerce systems – Odoo, Tryton)
* **Games** (e.g., Battlefield series, Sid Meier\’s Civilization IV…), **websites and services** (e.g., Dropbox, UBER, Pinterest, BuzzFeed…)



**2.3.3** **FEATURES PYTHON LANGUAGE**

**What is the Python Libraries ?**

We know that a module is a file with some Python code, and a package is a directory for sub packages and modules.

But the line between a package and a Python library is quite blurred.

A Python library is a reusable chunk of code that you may want to include in your programs/ projects.

Compared to languages like C++ or C, a Python libraries do not pertain to any specific context in Python.

Here, a ‘library’ loosely describes a collection of core modules.

Essentially, then, a library is a collection of modules. A package is a library that can be installed using a package manager like rubygems or npm.

**Python Standard Library**

The Python Standard Library is a collection of exact syntax, token, and semantics of Python.

It comes bundled with core Python distribution. We mentioned this when we began with an introduction.

It is written in C, and handles functionality like I/O and other core modules. All this functionality together makes Python the language it is.

More than 200 core modules sit at the heart of the standard library. This library ships with Python.

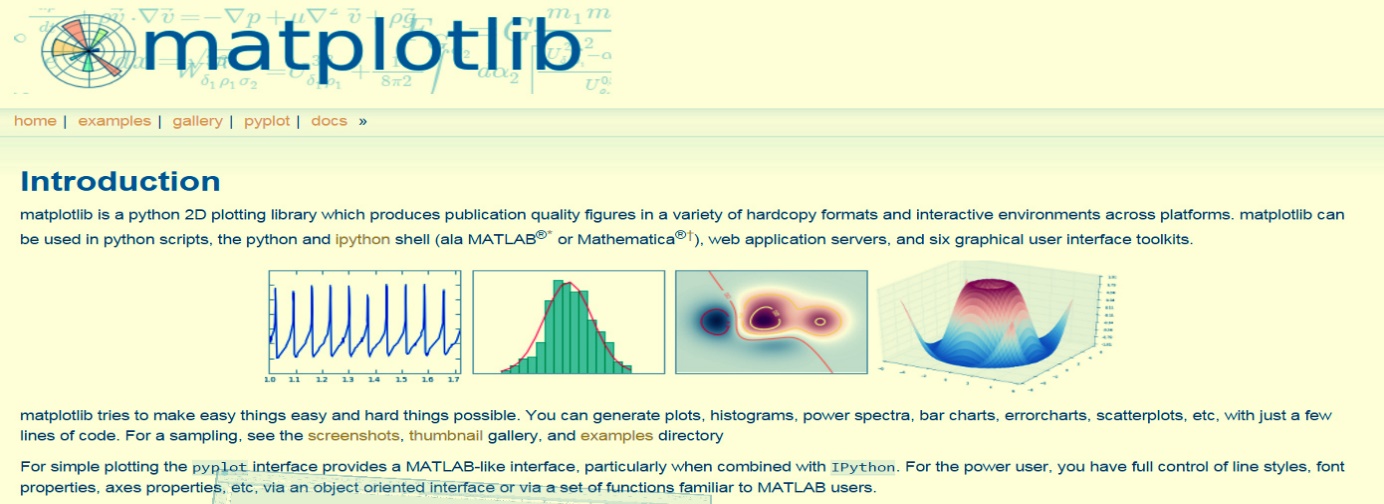
But in addition to this library, you can also access a growing collection of several thousand components from the Python Package Index (PyPI).

**Important Python Libraries**

Next, we will see Some Python libraries list that will take you places in your journey with Python.

**1. Matplotlib**

Matplotlib helps with data analyzing, and is a numerical plotting library. We talked about it in Python for Data Science.



**2. Pandas**

Like we’ve said before, Pandas is a must for data-science.

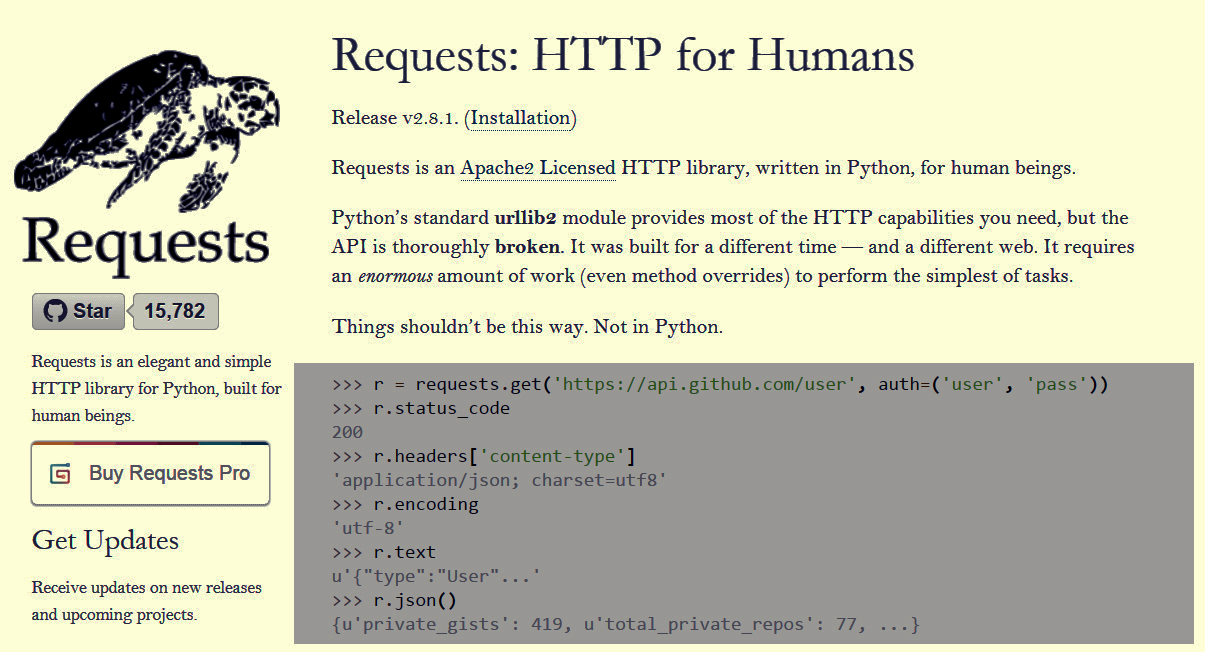
It provides fast, expressive, and flexible data structures to easily (and intuitively) work with structured (tabular, multidimensional, potentially heterogeneous) and time-series data.



**3. Requests**

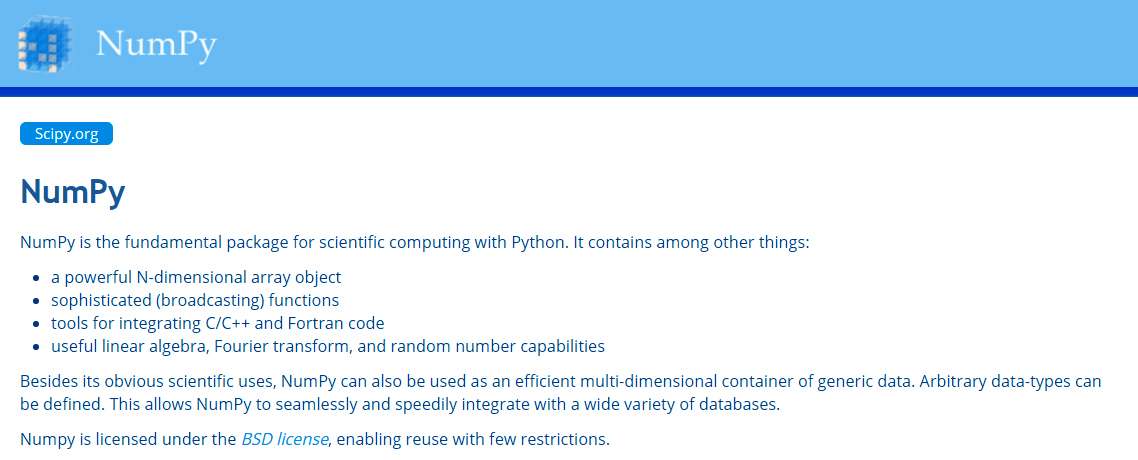
Requests is a Python Library that lets you send HTTP/1.1 requests, add headers, form data, multipart files, and parameters with simple Python dictionaries.

It also lets you access the response data in the same way.



**4. NumPy**

It has advanced math functions and a rudimentary scientific computing package.



There are many libraries in python. Few of them have been listed here

There are shown in the above figure.

**INTRODUCTION TO PROJECT**

Network automation using python Rest-API. We are building the infrastructure using network automation . we are going to create a controller that is APIC Controller (APPLICATION POLICY INFRASTRUCTURE CONTROLLER). Then you may think What is APIC Controller? APIC is controller that is shared service

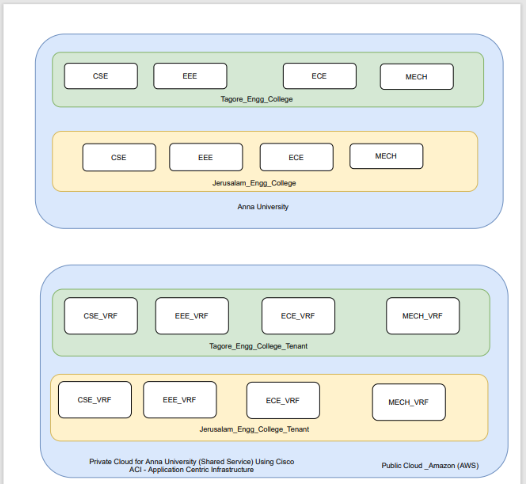
we can built it separate. We can provide the network infrastructure to our college, companies etc. Let’s say we are going to built infrastructure for our college like as “Tagore Engineering College” (or) “Jerusalem Engineering College”

Everybody can share the same infrastructure like Anna University. The whole fabric is shared by multiple users. You may have the doubt how to use it for free.

CISCO is providing a freesandboxes for people (or) users to practice and get familiar about the product, so that users can buy the product.

Practice the APIC controller using automation.

**SYSTEM DIAGRAM**



**Fig1 This is the Block diagram for all the college and departments for creating networks**

**(or)**

**ACI DIAGRAM**

**SYSTEM DESIGN**

**What is Cisco APIC**

The **Cisco** Application Policy Infrastructure **Controller** (**Cisco APIC**) is the main architectural component of the **Cisco ACI** solution. It is the unified point of automation and management for the **Cisco ACI** fabric, policy enforcement, and health monitoring.

In 2013, Cisco released their Software Defined Networking (SDN) solution for the data center known as Cisco Application Centric Infrastructure (ACI). For many years, the networking industry has been asking for an approach to configuring networking devices more efficiently than having to individually configure each and every router and switch.

Cisco’s release of ACI is the SDN solution to automate networking moves, additions and changes in the data center. The adoption rate of ACI in the industry in the first four years has been quite impressive.

An ACI implementation consists of the following three types of hardware components that work together to create the entire ACI fabric:

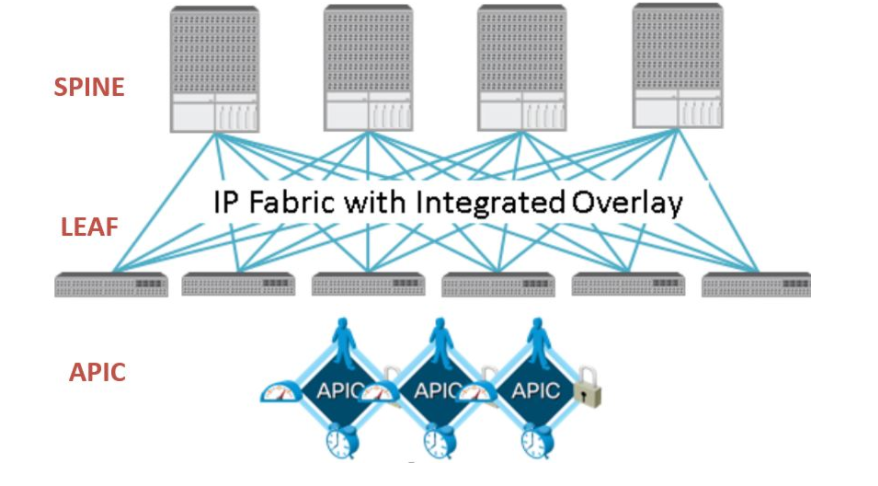
* 1.Spines: A group of Cisco Nexus 9000 switches work together to connect to all the leafs.

* 2. Leafs: As many as 300 Nexus 9000 switches are cabled to the spines as uplinks. The leafs also provide all connectivity into the ACI fabric for servers, bare metal or hypervisors, firewalls, load balancers and other routers and switches for external connectivity.

* 3. APIC: The Cisco Application Policy Infrastructure Controller (APIC) is the central brain for any ACI solution. The APICs are three or five servers in a fault tolerant cluster that provide the point of management, network performance monitoring and automate network provisioning.

**Goal:**

“Configuring physical and logical components manually takes time and can introduce errors. With Cisco ACI, we can define groups, interfaces, and policies once, and then push them everywhere from a central console. Once the scripts are  developed and the button is pushed, no human intervention is needed.”



**Fig 2: Block diagram of CISCO APIC**

**5.1 SYSTEM MAINTAINS & EVALUATION**

**TESTING & DEBUGGING**

Testing is the one of the most essential process that involve checking of the entire software files that helps in the proper functioning of the software. Testing provides a way of finding out the errors & faults that might have occurred during the development of the software. During the process of the testing of the software, all the necessary errors & faults that have occurred or arisen are traced & proper solutions regarding the errors. Are prepared.

Testing acts as an important phase of the software development life cycle.

Testing basically depends upon following two factors:-

* 1. Error Faults.
  2. Reliability.

Errors Faults defines the number of errors that have occurred during the development of the software that in effect have changed or diverted the entire process of the functioning of the software. The second important factor that comes out as result of testing is Reliability. By testing the entire software, we can easily find out the reliability of the software. So, testing provides a platform for the software developers to develop software that are error free and in effect, efficient and reliable.

Testing can be categorized into the following types depending on their use and

purpose in the development of the software. They are

1. **Functional Testing** Functional testing of the software comprises of testing of the function and modules that were created in the software and checking the accuracy of the functioning of the modules and functions that have been used in the proper and effective working of the software.
2. **Structural Testing** Structural Testing involves the process of testing the entire structure of the software that is developed (i.e. Logical as well as physical). All the logical steps related to the logical structure and the physical structure are tested for their accurate functioning and satisfaction for the developer.
3. **Debugging: -** Testing is the process of checking the errors, faults and failures that have occurred during the development or during the running of the system and which in turns have caused problems in the proper functioning of the system. Testing just provides a medium for searching out errors. On the other hand, de-bugging allows developers of the software to remove or make corrections on the errors that were found during testing of the software as de-bugging means, “removing of bugs”.

**SECURITY**

Security of the system means to define the safety of the software or the system. Software’s whether they may be large or small security play a vital role in improving the reliability and efficiency of the software. By securing the software at all the levels of the software makes the application as well as the software more reliable & efficient in working or functioning.

Security or safety of the system or the software can be done in many ways. These can be categorized as:-

 **Physical Security**

The term physical security of safety of the software, one means define the physical structure of the software safe and secure. This can be achieved by providing appropriate checks at each and every level of data entry and also by not giving any kind of structural changes

**Application Security**

In the software the text box will accept only those values for which it is placed in the project. Such as it the name of costumer is to be entered in the text then it will accept the only the string value and it will not accept the numeric values and if the user will give any wrong value then it will display the error message and if the user will give the string value in the numeric text box then also it will give the error message. In this way the application is secured for accepting the correct value in the database is secured.

**Administrator Security**

The administrator is the main of any system and if the user enter the password of the administrator then he/she will enter in administrator mode or only then he/she can change the administration password or add or remove login. Otherwise if any user login he/she can’t be able to access this administrator mode.

 **Operating Level Security**

This level of security provide a safety kit at times when any user free of worries and tension and tension of how the software may behave if they provide wrong input at any level of operating the software. For this several of checks are attached with the software of the software with on the Spot correction making etc.



**BENEFIT ANALYSIS**

When the estimates for a system developed, I need to consider several cost elements. Among them are hardware, personal, facility, operating and supply costs.

**Hardware Costs**

Hardware cost relates to the actual purchase or less of the computer and peripherals (for ex. Printer, disk drive, tape etc.). Determining the actual cost of the hardware is generally ore difficult when various users that for a dedicated stand alone system share the system. In some cases the best way to control for this cost is to treat it as an operating cost.

**Facility Cost**

Facility cost expenses include in the prevention of the physical site where the application or the computer will be in operation. This include wiring, flooring, acoustics, lightning and air cooling. These costs are treated as one time costs are incorporated into the overall cost estimate of the candidate system.

**Operating Cost**

This cost include all cost associated with the day-to-day operations of the System; the amount depends on the number of shifts, the nature of the applications and the caliber of the operating staff. There are various ways of covering operating cost. The amount Charged is based on computer time, staff time and volume of the output produced.

**Procedure for Cost & Benefit Determination**

Cost are incurred during throughout its life cycle. Benefits are realized in the form of reduced operating cost, improved operating image, staff efficiency or revenues. To what extent benefits outweigh costs is the function of cost/benefit analysis/ Cost benefit analysis is a procedure that gives a procedure of the various costs, benefit and rules associated with a system. The determination of costs and benefits entails the following step

**CONCLUSION AND FUTURE ENHANCEMENT**

**Fabric Scale and Other Enhancements**

The following table lists the new fabric scale and other enhancements features in this release:

**Table 3: New Software Features—Fabric Scale and Other Enhancements**

| **Feature** | **Description** | **Guidelines and Restrictions** |
| --- | --- | --- |
| Bookmarks | You can now bookmark almost any page, which enables you to go back to that page easily by choosing the bookmark from your list of bookmarks. In previous releases, this feature was represented as favorites (the star icon), and had less capability.  For more information, see the *Cisco APIC Getting Started Guide, Release 4.1(x)*. | None. |
| Confirmation and summary screens | Some of the wizards now include a confirmation screen and summary screen as the last steps. On the confirmation screen, you see a list of the policies that the wizard will create. You can change the names of the policies, if necessary. After the confirmation screen is the summary screen, which shows you the policies that the wizard created. You can no longer change the policies' names, but you can edit the properties of a policy. | None. |
| Default tab | This feature enables you to set a tab as the "favorite" on a page. Whenever you navigate to that page, that tab will be the default tab that is displayed. This feature is enabled only for the tabs in the Work pane.  For more information, see the *Cisco APIC Getting Started Guide, Release 4.1(x)*. | None. |
| Error counter enhancement | Physical interface configuration now includes error counter statistics information. | None. |
| Export tech support configuration data enhancement | This enhancement allows the user to export tech support data or configurations with read-only privileges.  For more information, see the *Cisco ACI Configuration Files: Import and Export*KB article. | None. |
| GTP load balancing | This feature enables the Cisco APIC to perform fabric load balancing based on GTP TEID.  For more information, see the *Cisco APIC Basic Configuration Guide, Release 4.1(x)*. | None. |
| Leaf switch uplink ports priority | When the fabric is scaled with numerous bridge domains, endpoint groups, and so on, and each are allocated a VLAN, this causes VLAN resource contention. Reloading a leaf switch in this state causes the leaf-to-spine switch uplink to enter the disabled state (those links do not come up). In this release, the leaf-to-spine switch uplinks are given a higher priority with the VLAN resource that is allocated to them, so that reloading a leaf switch while the switch is in a VLAN resource contention state does not affect the leaf-to-spine switch uplinks (the links come up). | None. |
| Multiple-context apps | You can now run an app in multiple GUI screens, or "contexts." For example, you can run the app while looking at a tenant's application profiles and while looking at the tenant's contracts. Prior to the 4.1 release, you could run an app only in one context; switching to a different context would close the app. | None. |
| New alerts | This release adds the following alerts:  Leaf x is Inactive: This alert warns you that a leaf switch became inactive, powered down, or disconnected.  New Switch Discovered: This informational alert informs you when a new switch is discovered.  Node Outage: Indicates that a node is either down or reloading.  Node x Must Be Reloaded: This alert warns you that an SSD must be reformatted and repartitioned.  OSPF Connectivity is Down: This alert warns you when OSPF connectivity is down. The alert lists the interfaces that have OSPF configured, but are not able to communicate with one another, and provides a recommended troubleshooting action.  Process Crash: This alert warns you that a process has crashed.  Split-Fabric Detected: Indicates that the fabric is split and that the controller is operating in read-only mode. | None. |
| Scale changes | This release includes the following scale changes:  ■      Maximum number of remote leaf switches: 128 (single pod)  ■      100 sub-interfaces per VRF and per L3Out  ■      4,000 MAC address EPGs | None. |
| Object Store Browser improvements | The Object Store Browser has the following improvements:  The Object Store Browser has a new, modernized look-and-feel.  You can now search by class, distinguished name, or URL, instead of only class and distinguished name. After you find an object, you can make the object a favorite, which enables you to go to your list of favorites and load the object from there.  You can now view the JSON response of your last query; previously you could only view the XML response.  The Object Store Browser by default displays all of the properties, even those that have no value. You can now hide the properties that do not have a value.  You can now navigate the distinguished name using the bread crumbs, which is simpler and easier to use.  You can now only view a distinguished name's stats, faults, or health if there is applicable data. |  |

**APPENDIX 1**

**SOURCE CODE:**

* 1. **FW\_GW\_project\_Master.py**

**import requests**

**import json**

**from string import Template**

**requests.urllib3.disable\_warnings()**

**class AuthenticationError(Exception):**

**pass**

**class Client:**

**def \_\_init\_\_(self, host, usr, pwd):**

**#self.jar = requests.cookies.RequestsCookieJar()**

**self.host = host**

**self.usr = usr**

**self.pwd = pwd**

**self.client = requests.Session()**

**#Pushing the configuration in the APIC controller**

**def POST(self, url, data,Role):**

**response= self.client.post('https://%s%s' % (self.host, url),data=json.dumps(data),timeout=5,verify=False)**

**resp=response.text**

**if 'error' in resp:**

**print("\n!!!!{}: Config already exist or config issue..Code{}\n".format(Role,response))**

**print("!!!!Error code:{}\n".format(resp))**

**#raise AuthenticationError**

**else:**

**print(">>>>{}:>>>>>Done # Status Code>{}".format(Role,response))**

**return response**

**#Login into APIC Controller using static Credentials.**

**def login(self):**

**data = {"aaaUser": {"attributes": {"name": self.usr, "pwd": self.pwd}}}**

**res= self.client.post('https://%s/api/aaaLogin.json' % (self.host),data=json.dumps(data),timeout=5,verify=False)**

**print(res)**

**if res.status\_code != 200 or 'error' in res.json()['imdata'][0]:**

**raise AuthenticationError**

**#T1:Create Tenant,VRF**

**def tenant(self,Tname,VRF):**

**Role='T1:tenant:{},VRF:{}'.format(Tname,VRF)**

**print("\nCreating the tenant:{}\n".format(Tname))**

**data = { "fvTenant":{"attributes":{"dn":"uni/tn-"+Tname,"status":"created,modified"},"children":[**

**{"fvCtx":{"attributes":{"dn":"uni/tn-"+Tname+"/ctx-"+VRF+"\_vrf","name": VRF+"\_vrf","rn":"ctx-"+VRF+"\_vrf","status":"created,modified"},"children":[]**

**}}]}}**

**return self.POST('/api/mo/uni/tn-{}.json'.format(Tname), data,Role)**

**def main():**

**client.login()**

**if \_\_name\_\_ == '\_\_main\_\_':**

**main()**

**2. Prework.py**

**def main():  
    from FW\_GW\_project\_Master import Client  
    host = "**[**sandboxapicdc.cisco.com**](http://sandboxapicdc.cisco.com/)**"  
    usr="admin"  
    pwd = "ciscopsdt"  
    tn = input("Enter the Tenant Name:")  
    #tn = "Indhu\_tenant"  
    vrf = input("Enter the VRF Name:")  
    #vrf ="Indhu\_vrf"  
    ACTION = input("Are you sure you want to push the configuration (y/n): ")  
  
    if ACTION in ("y","yes","Y","YES"):  
     FABRIC=Client(host, usr, pwd)  # Define the class.  
     print("Calling the Master function -> Authenticating into the Controller")  
     FABRIC.login()   #Calling the (.)Member(function) of the class.    
     t1=FABRIC.tenant(tn,vrf)  
  
    elif ACTION in ("n","no","N","No"):  
      print("Ending the script")  
    else:  
      print("Please enter yes or no.")  
  
   
if \_\_name\_\_ == '\_\_main\_\_':  
    main()**

**C:\Users\gmohan\python\Indhu>python Prework.py  
Enter the Tenant Name: Tagore\_Engineering\_college\_  
Enter the VRF Name: CSE**

**Are you sure you want to push the configuration (y/n): y  
Calling the Master function -> Authenticating into the Controller  
<Response [200]>  
  
Creating the tenant:   
  
>>>>T1:tenant:Tagore\_Engineering\_college\_tn,VRF:CSE \_vrf:>>>>>Done # Status Code><Response [200]>  
  
C:\Users\gmohan\python\Indhu>**

**Cisco SDN APIC :**

[**Application Policy Infrastructure Controller**](https://www.cisco.com/c/en/us/products/cloud-systems-management/application-policy-infrastructure-controller-apic/index.html#:~:text=Cisco%20APIC%20is%20the%20main,and%20optimizes%20performance%20and%20agility.)

SDN: Software defined Networking. It's a Private Cloud Networking.

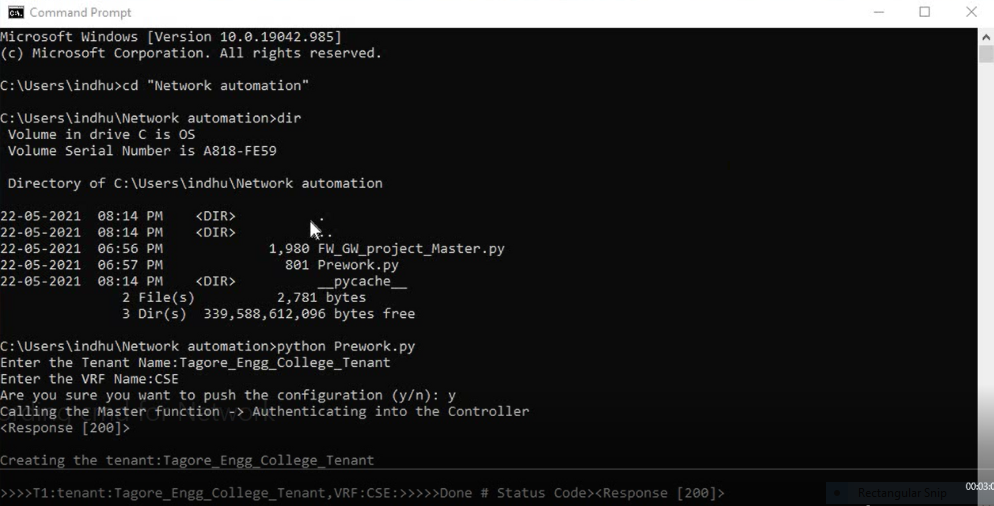
Public Cloud Network : Amazon, Google, Microsoft.

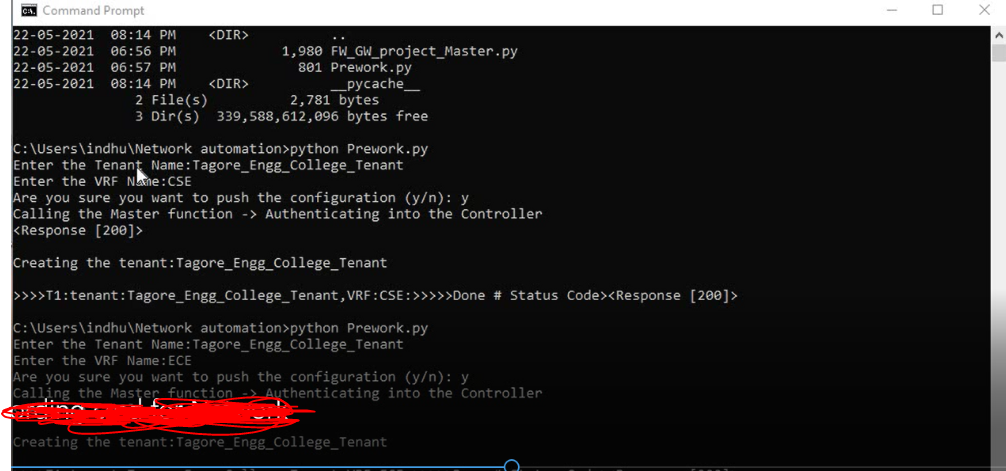
[https://sandboxapicdc.cisco.com](https://sandboxapicdc.cisco.com/)

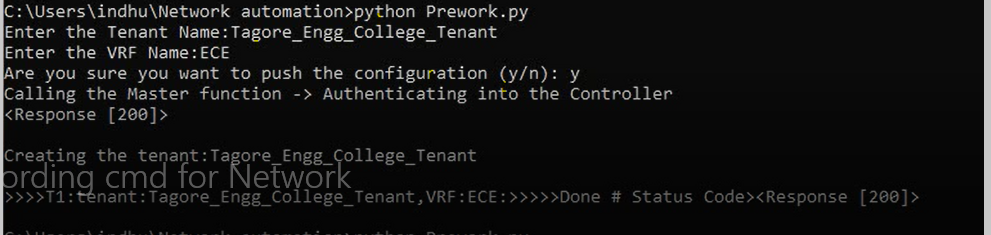
    host = "[sandboxapicdc.cisco.com](http://sandboxapicdc.cisco.com/)"  
    usr="admin"  
    pwd = "ciscopsdt"

**PYTHON SCRIPTING (or) COMMAND PROMPT**

**COMMAND PROMPT**







# **Automating Network Provisioning with Cisco APIC — Exploring the**

# **REST API**

**How to automate the configuration of network devices?**

[Cisco APIC-EM (= Application Policy Infrastructure Controller)](http://www.cisco.com/c/en/us/products/cloud-systems-management/application-policy-infrastructure-controller-apic/index.html) is a controller that can help us with that task. In this blog post, we will explore its modern REST API for accomplishing basic tasks like creating, reading, updating and deleting (CRUD) objects like

* APIC users (e.g. administrators)
* network discovery tasks
* network devices and their configurations.

**What is Cisco APIC?**

See [here a nice 4 minute Cisco Youtube commercial about APIC](https://www.youtube.com/watch?v=5YpAF9YG358). And [here](https://www.youtube.com/watch?v=Ixdpk2Hqcro&ebc=ANyPxKrGlEbVAUOYSKe4AotO6hiDD75Jub69XSE_oIrm_VFE7P3KzQu3nVH86_ond0darOil9dWU9tNjIMMsRcl4hIesF1fK5w) you find a [Cisco live!](http://www.ciscolive.com/global/) 40 minute session, which gives a short overview. I will summarize:

* APIC is a controller with Web Interface and REST API, which can be used to Create, Read, Update and Delete (CRUD) following kind of objects (and more)
  + Network Devices
    - Locations
    - Interfaces
      * Links
      * Hosts
  + Policies
  + Tasks
* Anything you can do in the Web interface can also be done via the REST API (similar to Ruby on Rails, RoR)
* The video is showing how a REST test tool like Chrome Postman or simple Python scripts can be used to interface with the REST API. Some examples are:
  + displaying a list of all network devices
  + displaying a the network path from arbitrary address A to arbitrary address B
  + finding an ACL on a network path that is blocking a certain application

**Cisco SDN APIC :**

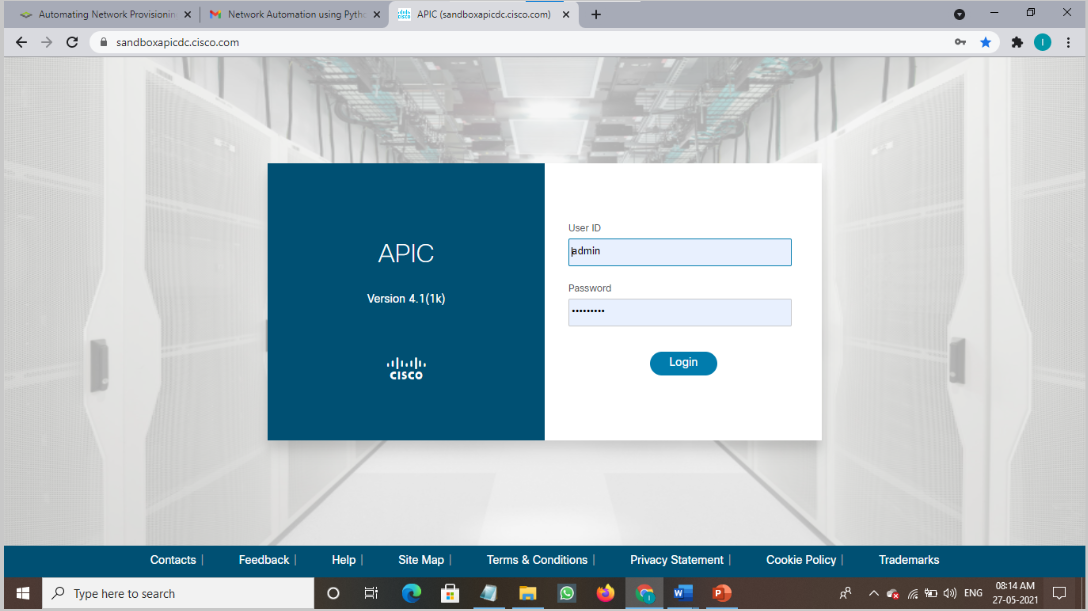
[**Application Policy Infrastructure Controller**](https://www.cisco.com/c/en/us/products/cloud-systems-management/application-policy-infrastructure-controller-apic/index.html#:~:text=Cisco%20APIC%20is%20the%20main,and%20optimizes%20performance%20and%20agility.)

SDN: Software defined Networking. It's a Private Cloud Networking.

Public Cloud Network : Amazon, Google, Microsoft.

[https://sandboxapicdc.cisco.com](https://sandboxapicdc.cisco.com/)

    host = "[sandboxapicdc.cisco.com](http://sandboxapicdc.cisco.com/)"  
    usr="admin"  
    pwd = "ciscopsdt"



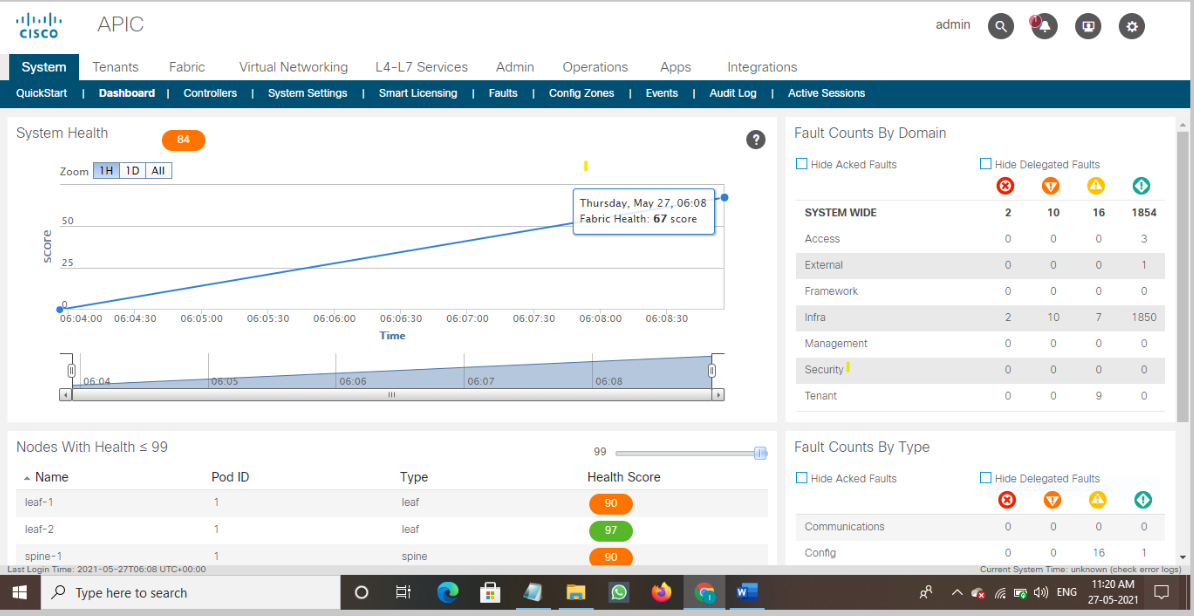
**Fig 3 APIC CONTROLLER**

**Why should you install Cisco APIC?**

The installation on VMware 5.1 or 5.5 would require [6 vCPUs and 64 GB of RAM, 500 GB disk and 200 MB disk I/O speed](http://www.cisco.com/c/en/us/td/docs/cloud-systems-management/application-policy-infrastructure-controller-enterprise-module/1-0-x/release-notes/apic-em_1-0-0_rn.pdf). A monster of an application. Wow.

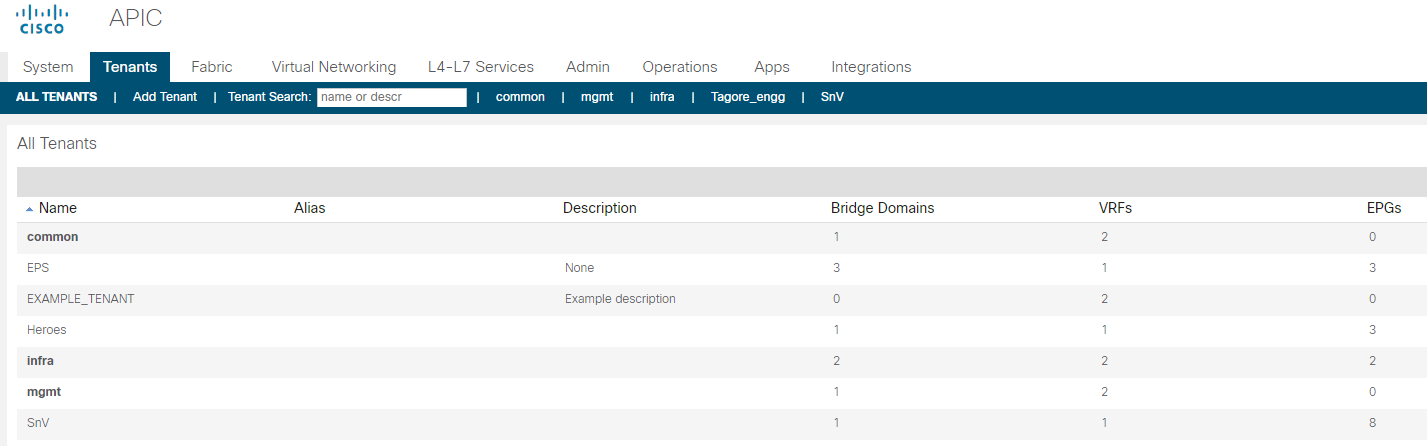
To be more specific: you can install Cisco APIC, but you do not need to install Cisco APIC, if you just want to explore the Cisco APIC interfaces: there is a more clever, cloud-based possibility: [DevNet sandboxes](https://devnetsandbox.cisco.com/RM/Topology/" \t "_blank) offered by Cisco.

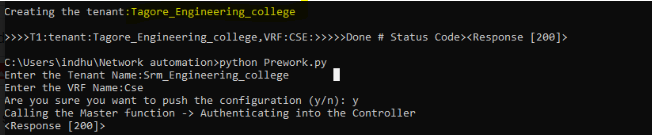
If you still want to install it, the SW can be found via [this DevNet page](http://developer.cisco.com/site/apic-em/) (use IE or Firefox; since Chrome is not supported; click on „3“). The SW is also available on Cisco’s [SW repository](https://software.cisco.com/download/release.html?mdfid=285968390&softwareid=286278832&release=1.2(2h)&relind=AVAILABLE&rellifecycle=&reltype=latest). However, authorization is required there, while in DevNet, the access is less restrictive.



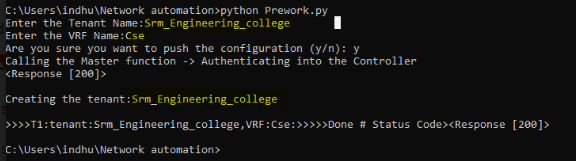
**Fig: Dasboard of APIC controller**

**Build Network using Python Script :**





**Fig**Building network for Tagore Engineering college and pushing the configuring



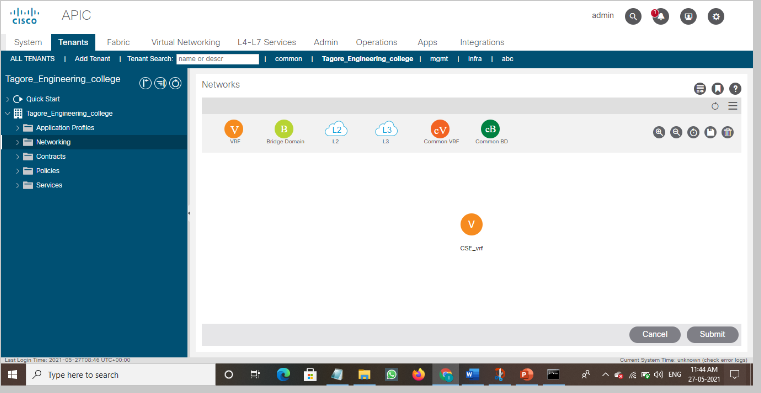
**Fig** Building network for SRM Engineering college and pushing the configuring

**Tenants are added :**



**Fig : Both the tenants are added to network**

**DEPARTMENT HAS BEEN CREATED :**

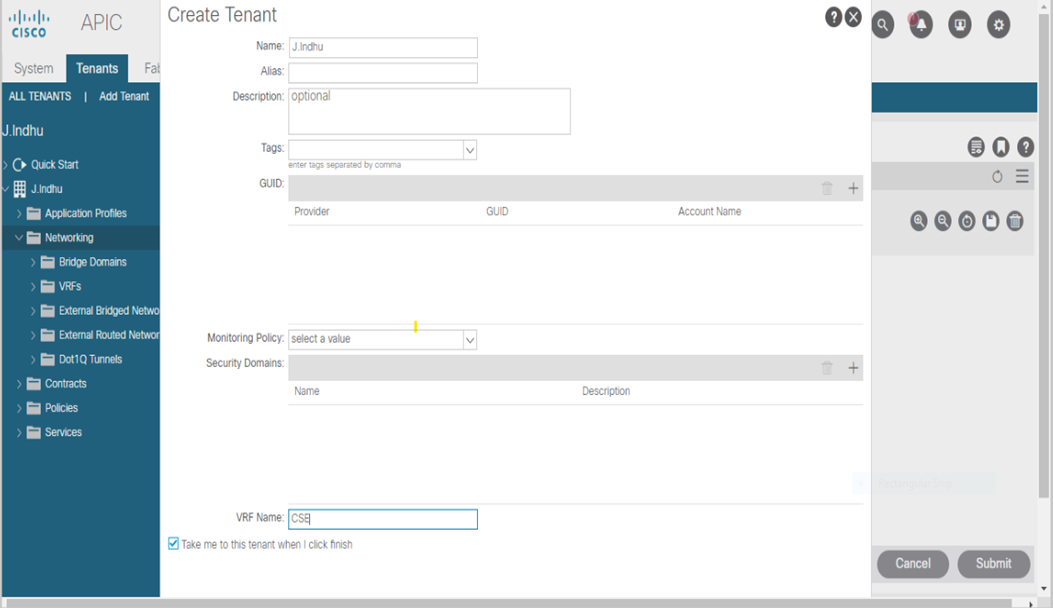


**Fig:We have built department for our college Tagore Engineering college**

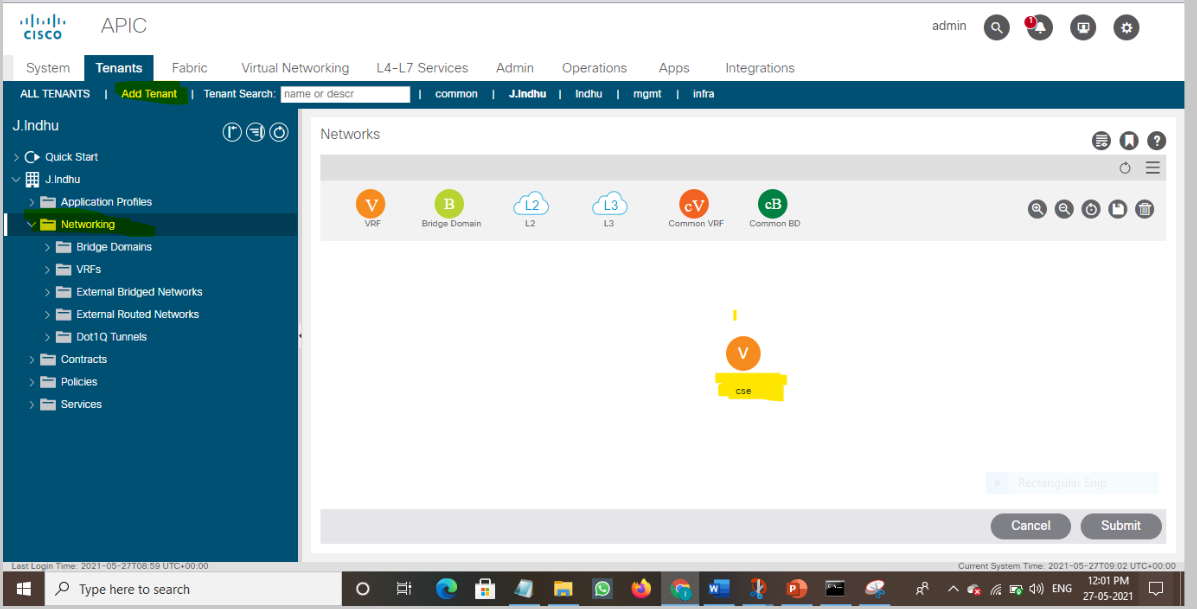


**Fig: Both tenants are added to the network**

**WE CAN CREATE IT MANUALLY ALSO :**

****

**Fig: Can create it manually**



**Fig: Manually created with Tenanat name and VRF**

**USERS ACCESS**

**Step1 start**

**Step2 First step is to enter the APIC controller**

**Step3 With the help of this url enter the sandbox** [**https://sandboxapicdc.cisco.com/**](https://sandboxapicdc.cisco.com/)

**Step4 And entering the website should begin with Login credentials**

**Step 5 By**   host = "[sandboxapicdc.cisco.com](http://sandboxapicdc.cisco.com/)"

usr="admin"  
    pwd = "ciscopsdt"

**Step6 Then finally we get in cisco Apic Controller**

**Step7 With python scripting we can see the option<Tenant>**

**Step8 <Tenant> and you can see many options**

**Step9 Choose networking option**

**Step10 we have builted the infrastructure for college**

**CONCLUSION**

**WORK DONE “NETWORK AUTOMATION USING PYTHON RESTAPI”** was successfully designed and is tested for accuracy and quality .During this project we have accomplished all the objectives and this project meets the needs of the organization, The developed will be used in searching, retrieving and generating information for the concerned requests.