**POD Process workflow**

Graphical user interface, application, Teams

Description automatically generated

**Requests need to raise :**

<https://carms.nsiam.vzwnet.com/>

1. EDN workstation access

i. EDN Workstation Access

ii. EDN Account for Application Access

2. <https://webtools.verizon.com/> >> ADOM >> ADOM GROUP SEARCH

3. Atoll user access

SDLC\_GYDV\_GIT\_DEVELOPER

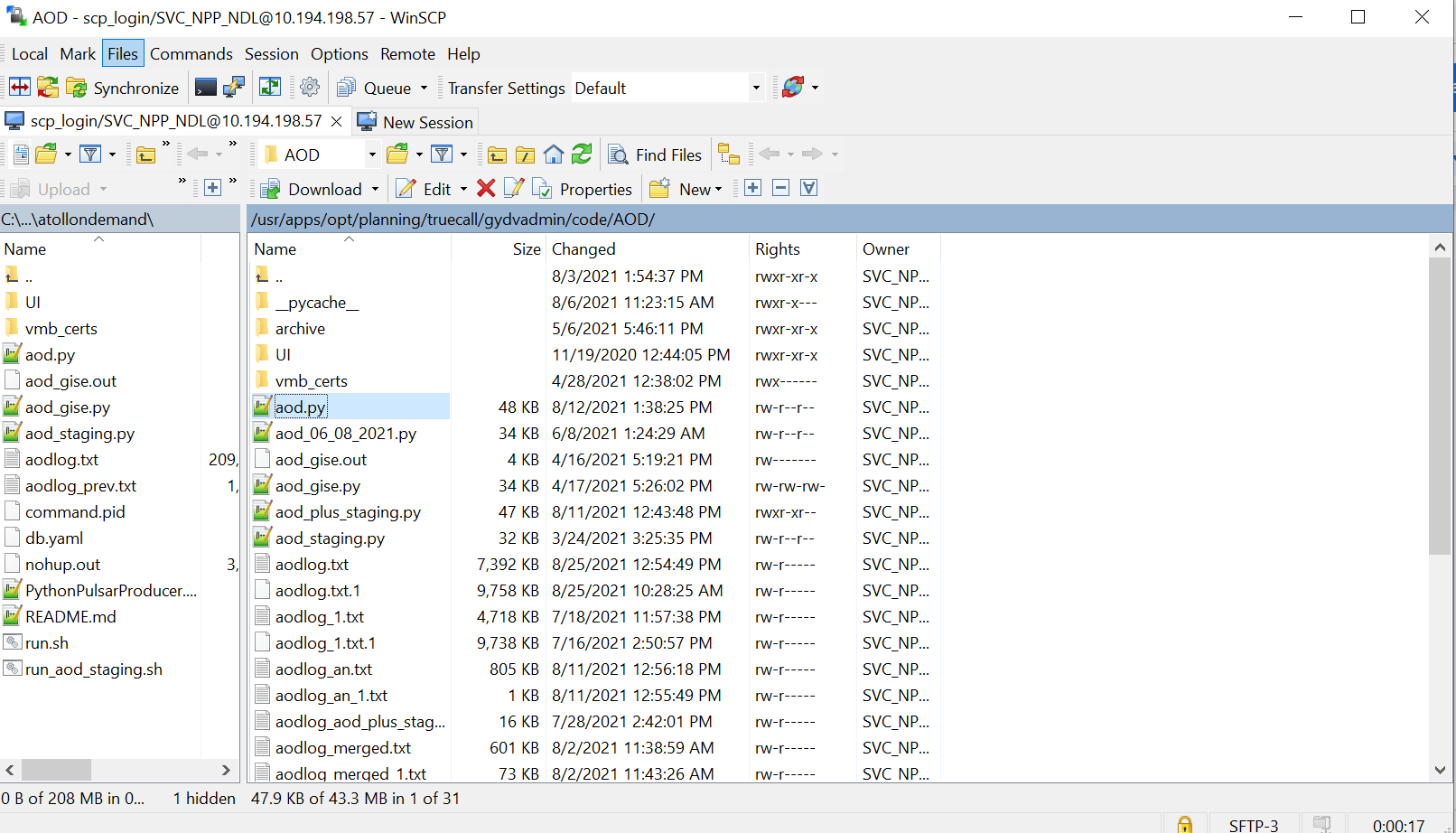
4. Atoll tickets are created at this location

https://atyourservice.verizon.com/ays?id=support HYPERLINK "https://atyourservice.verizon.com/ays?id=support&ci=6e88edf4db4a53487e09feb5ae961988&kb\_number=KB0030229 --> Create An Incident

| Projects | Git lab repo |
| --- | --- |
| Atoll on demand UI | https://gitlab.verizon.com/Network\_Planning\_Platform/atollondemand |
| Atoll on demand backend | https://gitlab.verizon.com/Network\_Planning\_Platform/atoll-on-demand-backend-engine |

Atoll on demand UI Server : <http://10.194.198.57:10533/atoll/>

To start the atoll UI server : Clone the repo to the linux server machine using winscp to [10.194.198.57](http://10.194.198.57:10533/atoll/), then check the port and host in aod.py and run ./run.sh in the command prompt. You can see server will be running [http://10.194.198.57:10533/](http://10.194.198.57:10533/atoll/) and log will be written in aodlog.txt



Atoll on demand backend : ondemand.py [ will be running in Virtual boxes ]

If multiple developers are working on test server which is as of now **txslatollpa72v.win.eng.vzwnet.com** in their respective workspaces

Atoll can be run by the command python ondemand.py --test

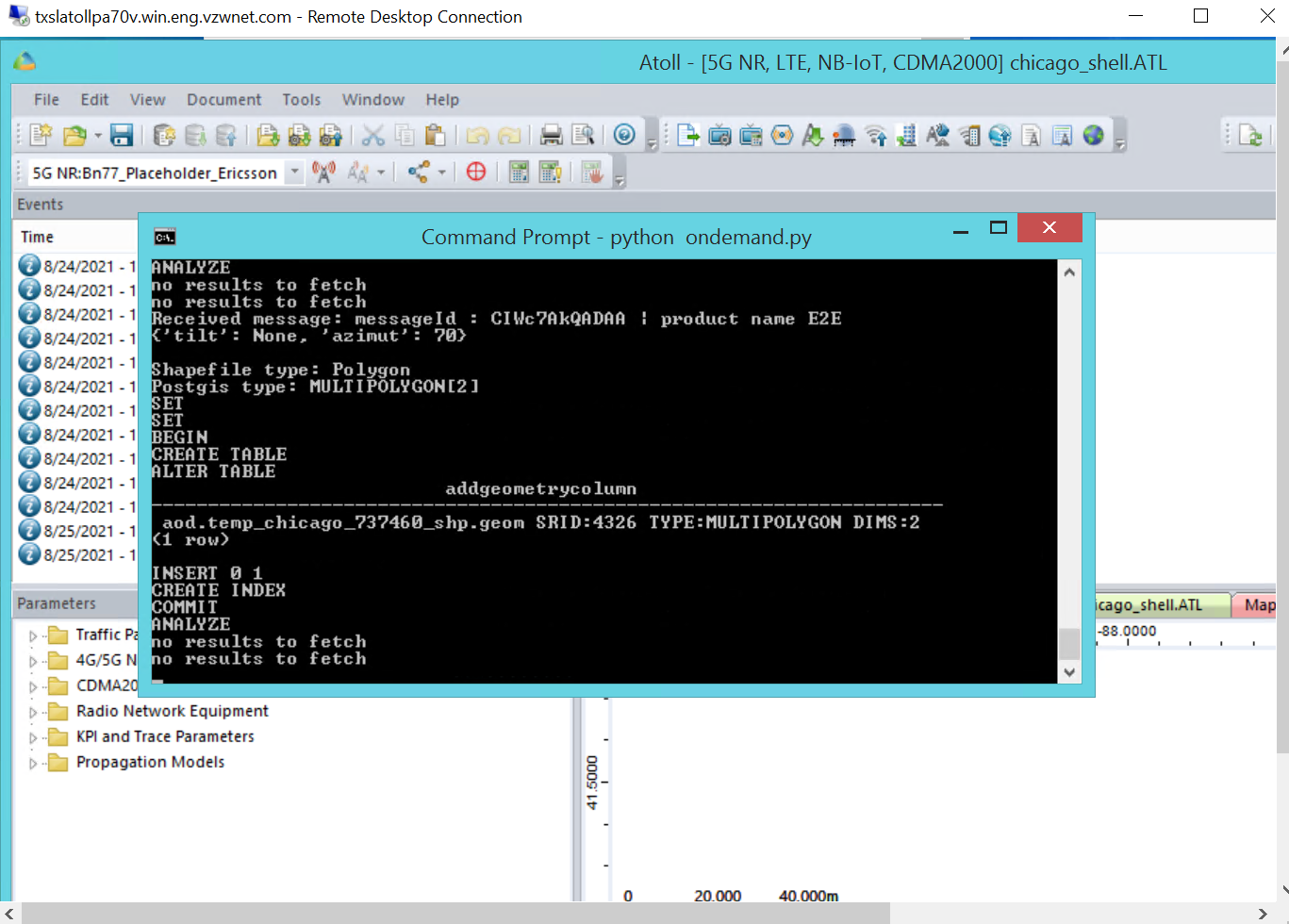
in that way atoll opens in the respective developer workspace instead of the server itself

Virtual boxes Computer names :

| Computer name | User name |
| --- | --- |
| txslatollpa70v.win.eng.vzwnet.com | win-vzwnet\username product\_name\_queue\_custom |
| txslatollpa71v.win.eng.vzwnet.com | win-vzwnet\username |
| txslatollpa72v.win.eng.vzwnet.com | win-vzwnet\username - product\_name\_queue\_test, |
| txslatollpa73v.win.eng.vzwnet.com | win-vzwnet\username |
| txslatollpa74v.win.eng.vzwnet.com | win-vzwnet\username |
| txslatollpa75v.win.eng.vzwnet.com | win-vzwnet\username |
| txslatollpa76v.win.eng.vzwnet.com | win-vzwnet\username |
| txslatollpa77v.win.eng.vzwnet.com | win-vzwnet\username |
| txslatollpa78v.win.eng.vzwnet.com | win-vzwnet\username |
| txslatollpa79v.win.eng.vzwnet.com | win-vzwnet\username |
| Txslatollpa80v.win.eng.vzwnet.com | win-vzwnet\username |
| Txslatollpa81v.win.eng.vzwnet.com | win-vzwnet\username |
| Txslatollpa82v.win.eng.vzwnet.com | win-vzwnet\username |
| Txslatollpa83v.win.eng.vzwnet.com | win-vzwnet\username |
| Txslatollpa84v.win.eng.vzwnet.com | win-vzwnet\username |
| Txslatollpa85v.win.eng.vzwnet.com | win-vzwnet\username |

**Process to login to VM box:**

1. Request for Remote Desktop Manager from <https://webtools.verizon.com/>
2. Login to Remote Desktop (RDP) by giving any of the above Virtual box name , username and password.
3. You can see in one of the command prompt ondemand.py will be running.



**Atoll on Demand API end points :**

1. <http://10.194.198.57:10534/atoll/coverage_request>
2. <http://10.194.198.57:10534/atoll/coveragerepeater_request>
3. <http://10.194.198.57:10534/atoll/coverage_request>
4. http://10.194.198.57:10534/atoll/coverage\_status
5. <http://10.194.198.57:10534/atoll/atoll_methods>
6. <http://10.194.198.57:10534/atoll/coverage_refresh>
7. <http://10.194.198.57:10534/atoll/coverage_sitetxrequest>
8. <http://10.194.198.57:10534/atoll/coverage_reqthreshold>
9. <http://10.194.198.57:10534/atoll/coverage_bandrequest>
10. <http://10.194.198.57:10534/atoll/stage/coverage_request>
11. <http://10.194.198.57:10534/atoll/stage/coveragerepeater_request>

**Database Details:**

| **Database** | **Details** |
| --- | --- |
| **Nppcoverages (PostgreSQL) - Prod** | **For credentials refer configs.py file in gitlab link** |
| **NPP database (Oracle) - Prod** | **For credentials refer configs.py file in gitlab link** |
| **NDL database (Oracle) - Staging** | **For credentials refer configs.py file in gitlab link** |
| **GWYVLATI – for** shp2pgsql -shapefile upload  **(GIS data)** | **For credentials refer configs.py file in gitlab link** |
|  |  |

**AOD Process:**

**Step 1 :**

Starts from requesting the coverage data by hitting the url in postman: <http://10.194.198.57:10534/atoll/coverage_request>

With input sample data (post url with json data)

**[**

**{**

**"tx\_id":"013384 01 B5\_10MHz\_2560 RAINBOW (2-0000)",**

**"site": "0384 RAINBOW (2-0000)",**

**"market":"saltlake",**

**"request\_type":"PSD",**

**"request\_subtype":"",**

**"receiver\_height":"5",**

**"user\_id":"",**

**"output\_request\_list":[**

**{**

**"type":"Vector",**

**"SubType":".shp",**

**"Threshold":"-104",**

**"Legend":""**

**}**

**],**

**"modified\_attributes":{**

**"XGTRANSMITTERS": {**

**"CALC\_RADIUS": "64373",**

**"CALC\_RADIUS2":"241000",**

**"CALC\_RESOLUTION":"30",**

**"CALC\_RESOLUTION2":"90",**

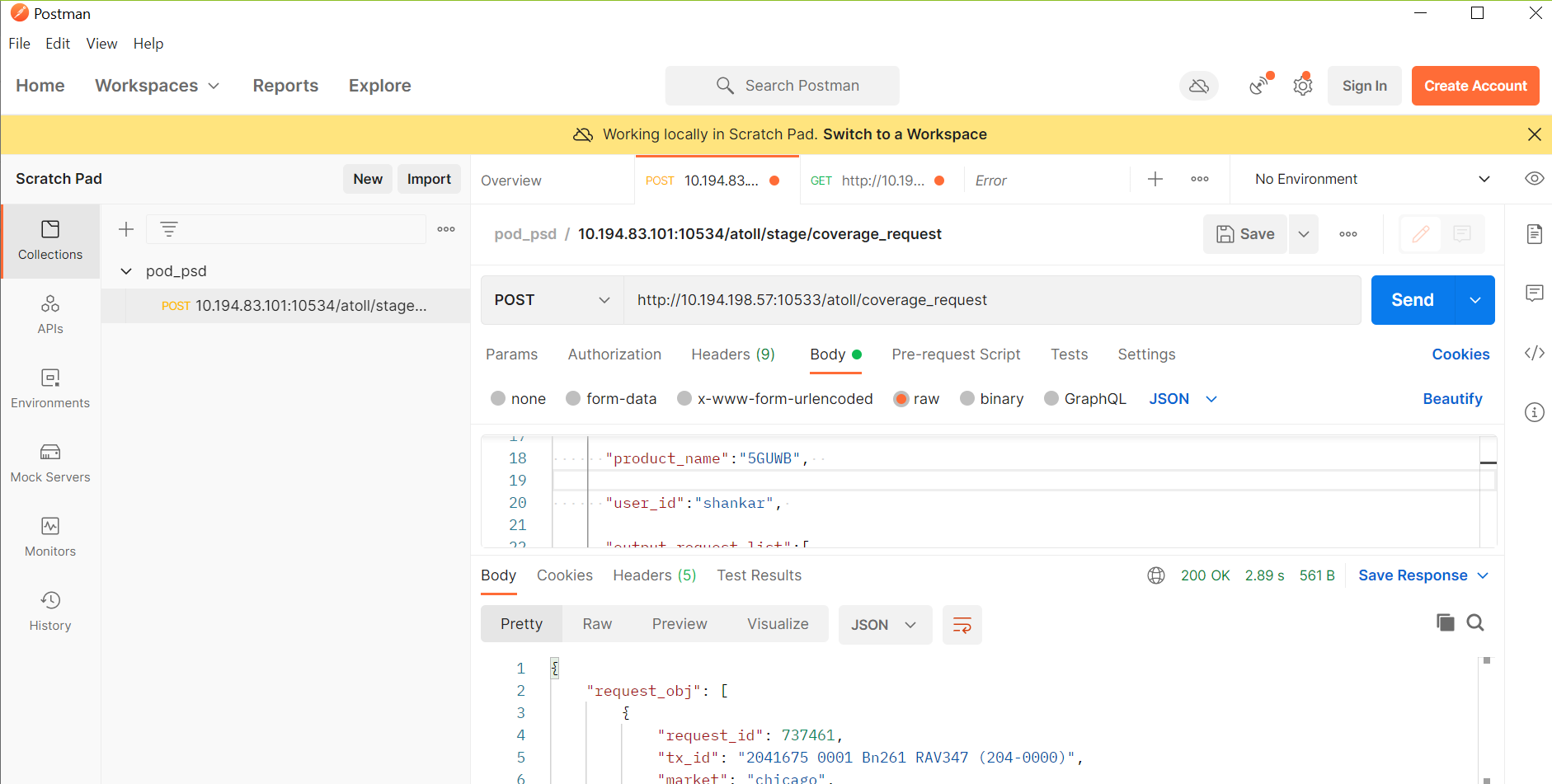
**"MAX\_RANGE":"0"**

**}**

**}**

**}**

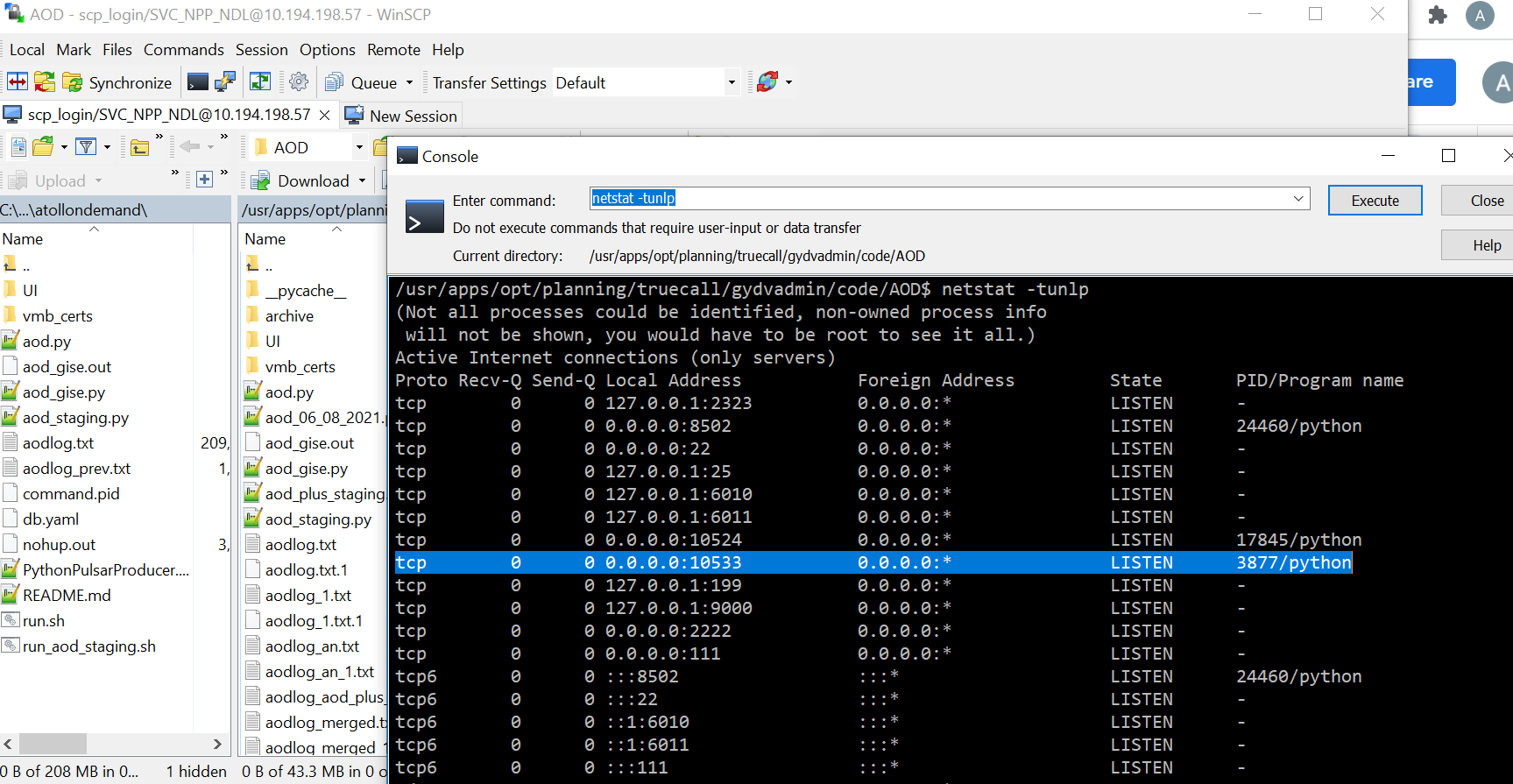
**]**



**Step 2:**

The link which we hit will be running from the server aod.py in the Linux machine with ip 10.194.198.57 and port 10533.

You can SCP to the Linux machine with your username and password (Should get from POD team) and see the server running there



Technical details :

That link will call the function coverage\_request() in aod.py which takes the input payload and extracts the required fields from payload and queries **(Q1)**  the NDL database and fetches other details for respective product names given in the payload. Product names - license\_management\_test, PSD\_test, 5GUWB, ANTENNA\_LOSS, LTE\_test, DODC\_test, DODC, UWB, E2E, pod-test, test, LTE

**Query (Q1):**

select Query:SELECT request\_id,tx\_id, market, site, request\_type, request\_subtype, status, cast(end\_time-start\_time AS VARCHAR) as proc\_time from aod.transmitter\_coverage\_request where request\_id = %s;

get sites:SELECT \* FROM CAROLINAS.SITES where name = '0483 ASHEBORO64WEST (STA-0000)'

get tx:SELECT \* FROM CAROLINAS.XGTRANSMITTERS x WHERE tx\_id = '0483 0001-01 Bn5-B5 ASHEBORO64WEST (R-0000)'

get\_cell\_LTE:SELECT \* FROM CAROLINAS.XGCELLSLTE x WHERE tx\_id = '0483 0001-01 Bn5-B5 ASHEBORO64WEST (R-0000)'

get\_cell\_NR:SELECT \* FROM CAROLINAS.XGCELLS5GNR x WHERE tx\_id = '0483 0001-01 Bn5-B5 ASHEBORO64WEST (R-0000)'

**Step 3:**

Then inserts **(Q2)** and update the record in NPP database with status “SENT TO ATOLL / SENT TO ATOLL TEST” and returns the json output as shown below for above sample payload and logs the details in aodlog.txt where the server is running.

**{**

**"request\_obj": [**

**{**

**"request\_id": 731117,**

**"tx\_id": "2041675 0001 Bn261 RAV347 (204-0000)",**

**"market": "chicago",**

**"site": "5580\_3 RAV347 (STA-0000)",**

**"request\_type": "public",**

**"request\_subtype": "",**

**"status": "SENT\_TO\_ATOLL",**

**"proc\_time": null,**

**"coverage\_output": []**

**}**

**],**

**"meta": {**

**"status": "ok"**

**}**

**}**

**Query(Q2):**

Establishes connection to NPP database and inserts the coverage request record

INSERTINTO aod.transmitter\_coverage\_request(request\_type,request\_subtype,market,site,tx\_id,user\_id,status,product\_name)values('RSRP','','ORLANDO','0081AMERICANSTAINLESS(STA-0000)','00810002-02Bn2-B2AMERICANSTAINLESS(2-0000)','FAILED\_TX','SENT\_TO\_ATOLL','LTE')returningrequest\_id;

Update Query:UPDATE aod.transmitter\_coverage\_request set atoll\_request\_object = (%s) where request\_id= (%s);

**Step 4: This is where backend code will be executed.**

Ondemand.py is the Python script which will be running in all the VM boxes which will continuously share the traffic (coverage request hits) across the VM boxes.

Which will executes to functions :

req\_prc.start() : Which will check the coverage\_requests **(Q3)**

calc\_prc.start() : Which will calculate the coverage data

req\_prc.start() :

Triggers the rest.py get\_request() function. It queries the aod.transmitter\_coverage\_request database and fetches the records with status “SENT\_TO\_ATOLL/FAILURE” with product names 'LTE', 'E2E', '5GUWB'. Fetches the required fields from the query results and passes to the MQ class function add\_to\_vmb.

**Query(Q3):**

**select \***

**from aod.transmitter\_coverage\_request**

**where**

**status in ('SENT\_TO\_ATOLL','FAILURE')**

**and product\_name in ('LTE', 'E2E', '5GUWB')**

**and atoll\_request\_object is not null**

**and attempts < 3**

**order by**

**priority desc, market**

**limit 50**

**for update**

**Step 5:**

Once the add\_to\_vmb function is called, it will verify the SSL with the given certs(ca.cert.pem, gydv.cert.pem, gydv.key-pk8.pem) and sends the websocket message to Atoll using the product name producer details in cpmap.py. Updates the coverage\_request record in NPPdatabase with status “ADDED\_TO\_QUEUE” **(Q4)**

**Query(Q4)**

**update**

**aod.transmitter\_coverage\_request**

**set**

**status = %(status)s,**

**{time} = %(timestamp)s,**

**comments = %(comments)s,**

**attempts = %(attempts)s**

**where**

**request\_id = {req\_id}""".format(req\_id = req\_id, time = time)**

**Step 6:**

Once the message is sent to ATOLL , calc\_prc.start() this function will start executing.

calc\_prc.start() :

This function triggers the Coverage.py script and calls the Coverage.run() function. Which will call the function receive\_from\_vmb(). It will receive the websocket messages(payload)/events with product name consumer details. Gets the request id and attempts from the event. Then updates the coverage\_request record with status “PROCESSING\_ATOLL” **(Q5)**

**Query 5**

**update**

**aod.transmitter\_coverage\_request**

**set**

**status = %(status)s,**

**{time} = %(timestamp)s,**

**comments = %(comments)s,**

**attempts = %(attempts)s**

**where**

**request\_id = {req\_id}""".format(req\_id = req\_id, time = time)**

**Step 7:**

Gets the active atoll objects and setup directories and paths for the object. Copies the atoll document and opens it. Extracts the required fields and adds the fields like site\_info, cells and repeater info to the document. Calculates the antenna loss using function get\_antenna\_loss() where all the mathematical calculations will be done. Sets threshold, properties, receiver height, interpolation parameters, loss model parameters, prediction resolution and calculates the coverage data using function calculate\_coverage(). Creates the vector folder using the function create\_geom() and exports the bill\_file using the function rssi\_psd\_create\_geom\_for\_transmitter() . Copies the billfile in the raster folder and returns the raster file using the persist\_raster() function. Then copies the raster file to EBS and updates the record with status “SUCCESS” **(Q6)**

**Query(Q7):**

**update**

**aod.transmitter\_coverage\_request**

**set**

**status = %(status)s,**

**{time} = %(timestamp)s,**

**comments = %(comments)s,**

**attempts = %(attempts)s**

**where**

**request\_id = {req\_id}""".format(req\_id = req\_id, time = time)**

**Step 8:**

When the user hits the api “/atoll/coverage\_status” with request\_id as payload, it will query the database with request id and return the json output of the calculated coverage data.

**Query(Q7):**

**"""SELECT status, comments from aod.transmitter\_coverage\_request where request\_id = {request\_id}; """.format(request\_id=request\_id)**