Overview

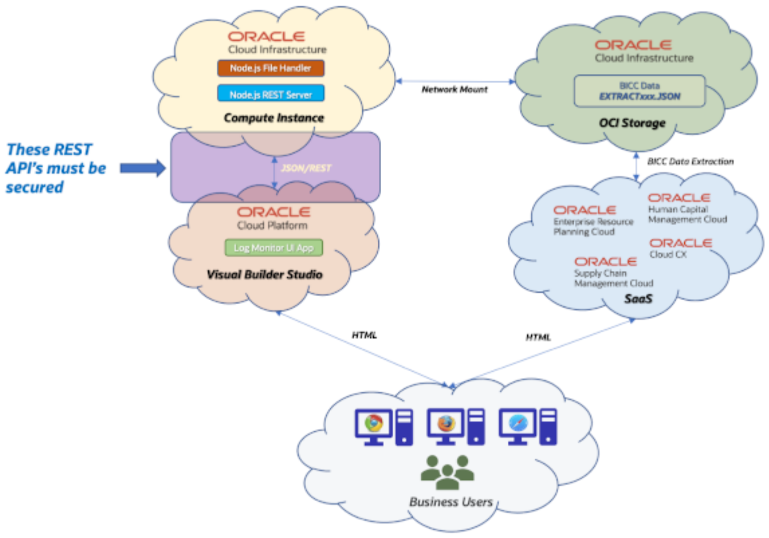
This article is a follow-up to a previous blog about **BICC Extract Logs Monitoring on OCI Native**. In that article I've described how OCI native services can be used to track the BICC status log files and how to provide these information via a set of REST API's for further external processing. The proposed architecture used a simplified architecture with no security features involved. Main reasons to keep the security out of scope of the previous article were the following as listed below:

- The intention was to keep the complexity of the previous article manageable.
- The provided code sample was intended to provide an ability to run generically without any dependencies at any device for evaluation and testing purposes - from local PC to to OCI Compute Instance
- Adding specific security features into the Javascript code itself has the risk of creating dependencies that might could require the need for rework in case security standards are changing, leaks being found or technologies are chaging
- Nevertheless its admited that we have to secure this solution via an alternate path - we have to avoid that the REST API's can be invoked by consumers which are not authenticated or via http traffic that is not encrypted
- As the original sample was introduced as a component in an OCI eco-system we will add the security features via embedding the solution into some additional OCI components in order to add the required functionalities

In the following sections we'll walk step by step through the various configuration activities after starting with an introduction into the proposed Security Architecture first.

Introducting a Security Architecture

In the architecture I've proposed in the previous blog we were using several OCI components that were consuming SaaS BICC log information and eventually providing s set of REST API's for being used by other applications. The Node.js and Epress.js program was focusing on the algorithm to read files, render the infomation according to the API structure and to provide features for a data refresh. The code does not include any security routines and is not using an https protocol either. The sample can run without any bigger efforts to configure an user acces, registration of certificates or similar. Its obvious that such a sample won't stand the security requirements of a real-life customers environment. We have two choices now: a) to add security functionality to the code with the risk of continous rework whenever standards are changing and also to blow-up the number of clode lines or b) to change the architecture in a way that these services can only be invoked in a secure way.

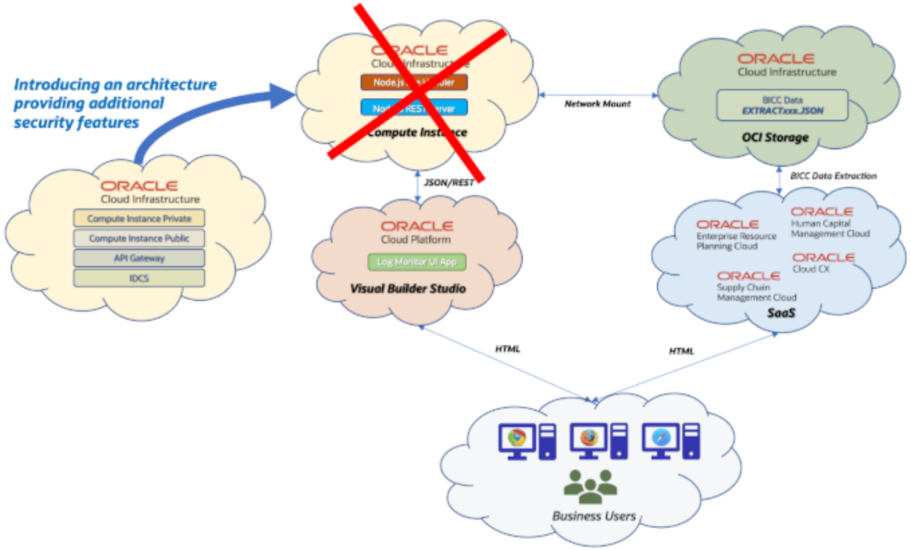


Need for an additional security layer

In the OCI landscape we have a couple of service offerings that will perfectly fit in here by using API Gateway for the hardening of the service access and IDCS for an embedding of these services into the security layer of OCI. Saying this, we must start fine-tuning the components formerly used as our Compute Instance that contains the Node.js and Koa.js application. Instead of running one single compute instance we will add more services here:

- One Compute Instance that has a public IP address and can be reached from the internet. This Linux VM will be used as a bastion server located in the same VNC as the other components
- Another Compute Instance, declared as private, will reside in the joined VNC, but jabe no public IP address. For maintenance operations like code updates, starting and stopping of services etc it can only be accessed from the bastion server
- API Gateway must be used for the deployment of public and secured REST API's. This deployment will reside in the same VNC as both Linux VM's and therfore have an access to the private REST API's. It will publish these services including all required security features.
- IDCS is always existent in an OCI environment and underlying the entire security architecture. We will create a custom application in IDCS to provide a controlled access to the REST services deployed created in API Gateway.

Users or services may get an access granted via IDCS and we can ensure that these API's work in a secure mode this way without adding any new codeline into the JS application or also without modifying the sources.

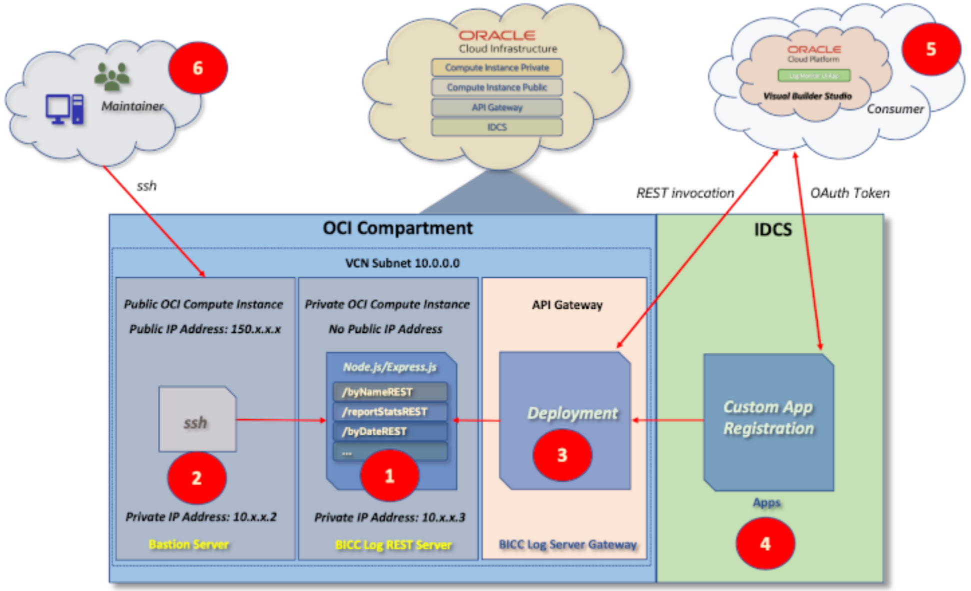


Required Architecture Changes

The components in this new architecture are shown below in detail:

1. The Private VM can't reached from outside the VCN and is meant to run the Koa.js and Node.js code. The REST API's are running strictly internal. This is a pre-condition to ensure these unprotected internal API's will be registered and mapped in API Gateway before published as protected REST services.
2. The Public VM is located in the same VCN as the Private VM, but has also an additional public IP address. Via this virtual machine the private machine is accessible for the maintenance of the software artefacts providing these REST API's. The role of this server can bee seen as those of a Bastion Host.
3. Every single REST API that is meant to be published, must be registered in an API Gateway deployment. This deployment itself will be hardened with a security setup that will be explained further down below in this article.

4. Once these API's were configured in API Gateway they become registered next within a custom application in IDCS. That will give access permission to groups and users in an OCI environment.
5. Users and groups with an access given via IDCS can consume these REST API's. Those will be primarily used by Tools like OIC or VBCS.
6. The maintenance operations like starting, stopping of services or software release management or bug fixing will be managed by a small group of experts who have access to the Public VM. The security setup for these maintainers will be kept independently from the security setup described above.

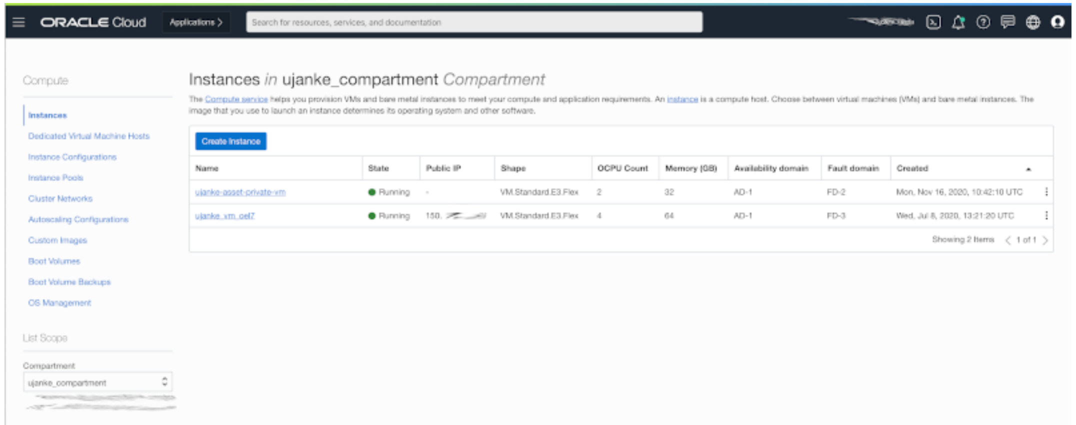


```
Details of an additional Security Layer
```

The configuration of this architecture will be explained more in detail below. As discussed this architecture will introduce a number of other cloud resources to segregate the duties for a security hardening of the application. This is seen as the better alternative in opposite to add security code into the application program itself. With Oracle OCI all these features are available depending on a customers cloud services subscriptions.

Configuring Public/Private Compute Cloud instances

As shown above in our architecture figure we need two Compute Cloud instances - one as a bastion host and another one to run the Node.js/Koa.js application. The screenshot below shows the two existing virtual machines - one with a public IP address and the other one remaining strictly internally.



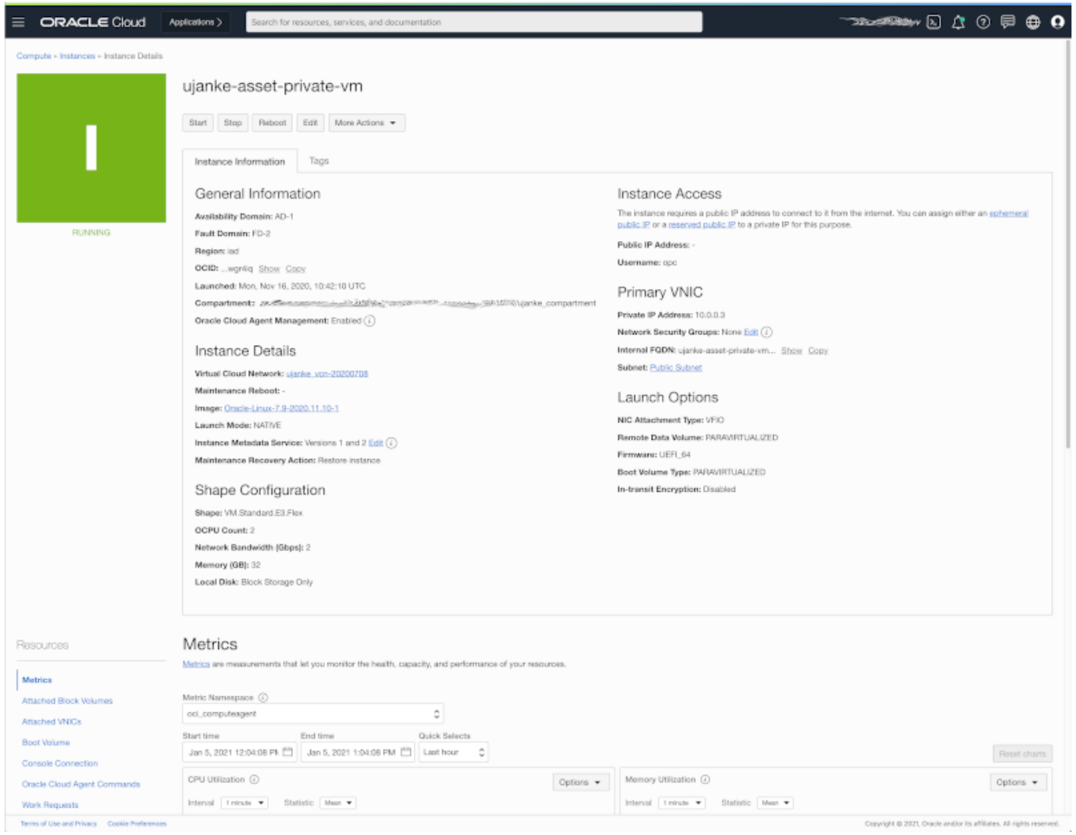
Instances *in* ujanke_compartment *Compartment*

The Compute service helps you provision VMs and bare metal instances to meet your compute and application requirements. An instance is a compute host. Choose between virtual machines (VMs) and bare metal instances. The image that you use to launch an instance determines its operating system and other software.

Name	State	Public IP	Shape	OCPU Count	Memory (GB)	Availability domain	Fault domain	Created
ujanke-asset-private-vm	● Running	-	VM.Standard.E3.Flex	2	32	AD-1	FD-2	Mon, Nov 16, 2020, 10:42:10 UTC
ujanke_vm_oel7	● Running	150. [redacted]	VM.Standard.E3.Flex	4	64	AD-1	FD-3	Wed, Jul 8, 2020, 13:21:20 UTC

Creating a Public and a Private VM

The private instance is responsible to cover a potentially high workload and should be sized accordingly. In our sample below we're good with an average sizing, but that might differ on the various customer use cases we can imagine. Here its a JS server that reads and parses BICC status log files and provides the consolidated data as REST API's.



ujanke-asset-private-vm

General Information

Availability Domain: AD-1

Fault Domain: FD-2

Region: iad

OCID: ...wgr4iq Show Copy

Launched: Mon, Nov 16, 2020, 10:42:10 UTC

Compartment: [redacted]ujanke_compartment

Oracle Cloud Agent Management: Enabled

Instance Details

Virtual Cloud Network: ujanke_vcn-20200708

Maintenance Reboot: -

Image: Oracle-Linux-7.9-2020.11.10-1

Launch Mode: NATIVE

Instance Metadata Service: Versions 1 and 2 Edit

Maintenance Recovery Action: Restore instance

Shape Configuration

Shape: VM.Standard.E3.Flex

OCPU Count: 2

Network Bandwidth (Gbps): 2

Memory (GB): 32

Local Disk: Block Storage Only

Instance Access

The instance requires a public IP address to connect to it from the internet. You can assign either an ephemeral public IP or a reserved public IP to a private IP for this purpose.

Public IP Address: -

Username: opc

Primary VNIC

Private IP Address: 10.0.0.3

Network Security Groups: None Edit

Internal FQDN: ujanke-asset-private-vm... Show Copy

Subnet: Public Subnet

Launch Options

NIC Attachment Type: VFIO

Remote Data Volume: PARAVIRTUALIZED

Firmware: UEFI_64

Boot Volume Type: PARAVIRTUALIZED

In-transit Encryption: Disabled

Metrics

Metrics are measurements that let you monitor the health, capacity, and performance of your resources.

Metric Namespace: oci_computeagent

Start time: Jan 5, 2021 12:04:08 PM

End time: Jan 5, 2021 1:04:08 PM

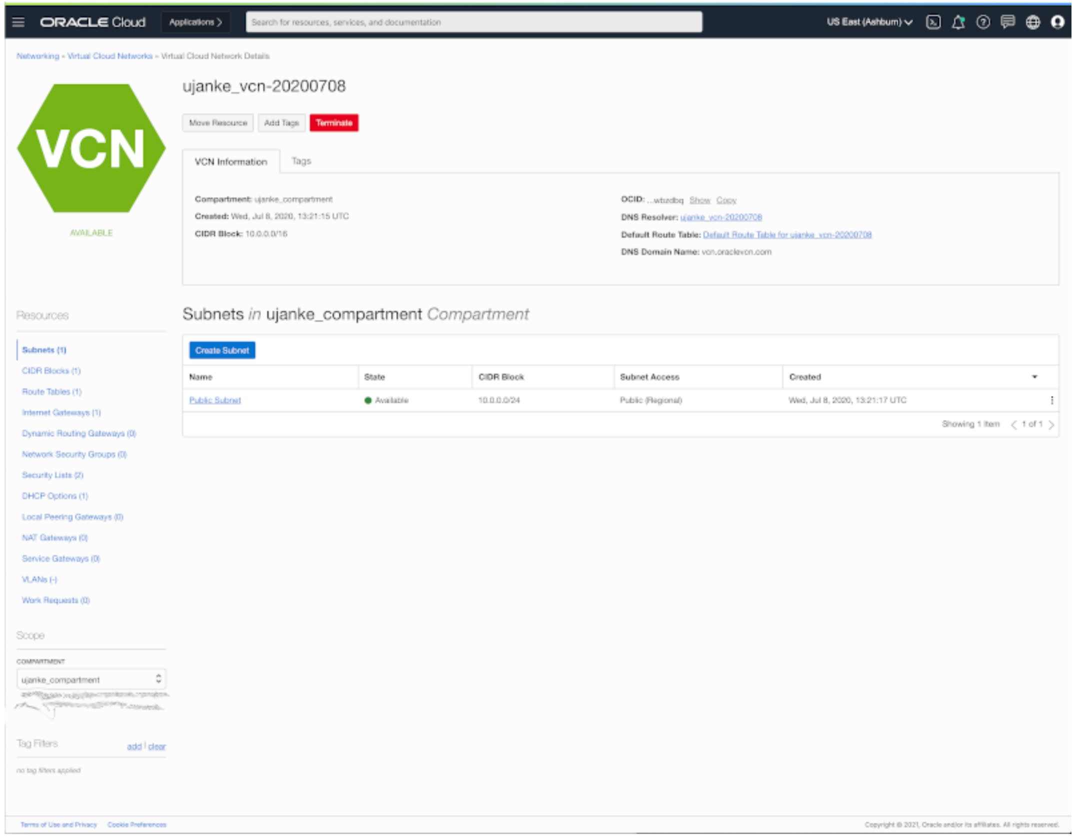
Quick Selects: Last hour

CPU Utilization — Interval: 1 minute — Statistic: Mean

Memory Utilization — Interval: 1 minute — Statistic: Mean

`Private VM Details`

As mentioned earlier on both instances belong to the same VCN configuration. In theory there could be two different VCN's being used with a correct and according routing setup. But as we will see later for the configuration of API Gateway it will be helpful if at least those both components exist in same VCN. Still: with an according routing setup it would also work if not using the same VCN.



Networking » Virtual Cloud Networks » Virtual Cloud Network Details

ujanke_vcn-20200708

Compartment: ujanke_compartment
Created: Wed, Jul 8, 2020, 13:21:15 UTC
CIDR Block: 10.0.0.0/16

OCID: ...wbzdbq Show Copy
DNS Resolver: ujanke_vcn-20200708
Default Route Table: Default Route Table for ujanke_vcn-20200708
DNS Domain Name: vcn.oraclevcn.com

Subnets *in* ujanke_compartment *Compartment*

Name	State	CIDR Block	Subnet Access	Created
Public Subnet	● Available	10.0.0.0/24	Public (Regional)	Wed, Jul 8, 2020, 13:21:17 UTC

Showing 1 Item < 1 of 1 >

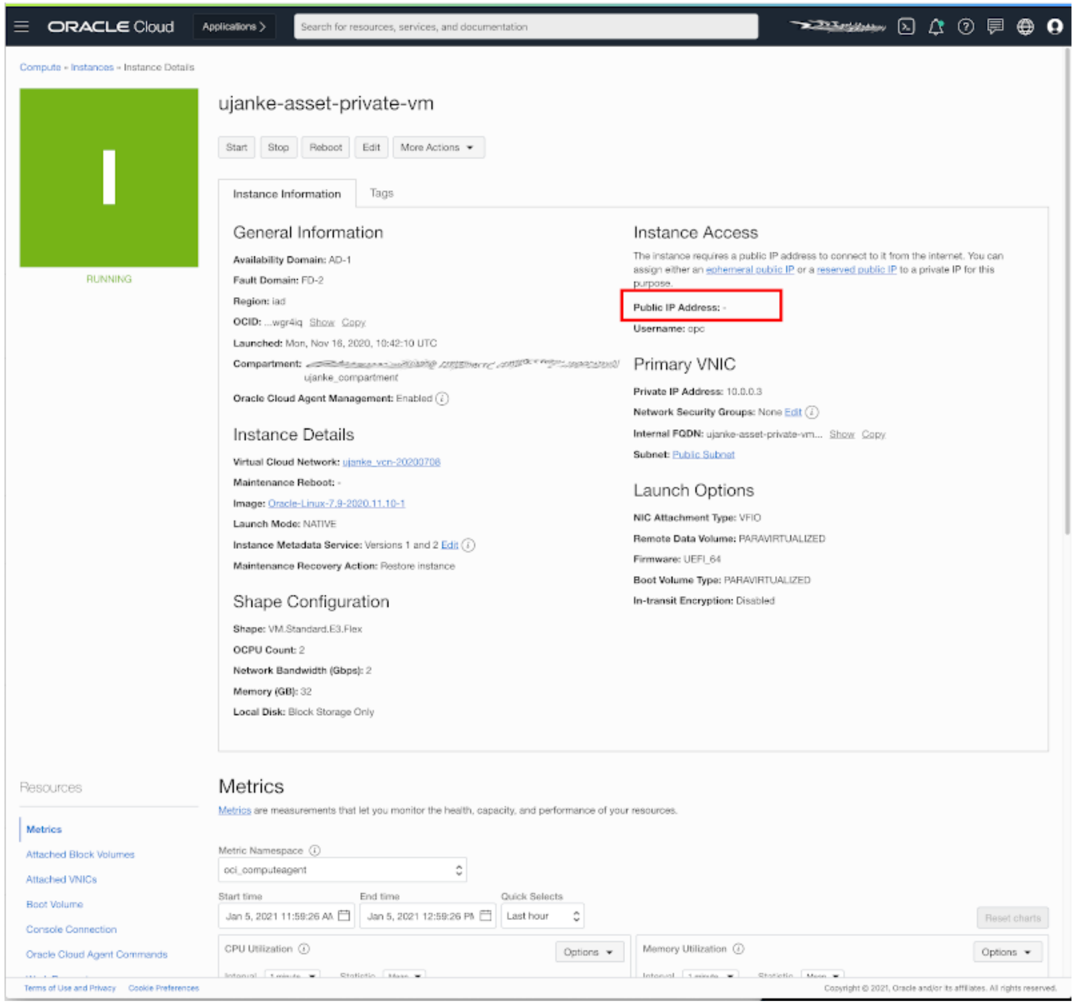
COMPARTMENT: ujanke_compartment

no tag filters applied

Copyright © 2021, Oracle and/or its affiliates. All rights reserved.

Public and Private VM sharing the same VCN

As shown below the Application hosting VM has no public IP address.



ujanke-asset-private-vm

General Information

Availability Domain: AD-1

Fault Domain: FD-2

Region: iad

OCID: ...wgr4iq Show Copy

Launched: Mon, Nov 16, 2020, 10:42:10 UTC

Compartment: [redacted] ujanke_compartment

Oracle Cloud Agent Management: Enabled

Instance Details

Virtual Cloud Network: ujanke_vcn-20200708

Maintenance Reboot: -

Image: Oracle-Linux-7.9-2020.11.10-1

Launch Mode: NATIVE

Instance Metadata Service: Versions 1 and 2 Edit

Maintenance Recovery Action: Restore instance

Shape Configuration

Shape: VM.Standard.E3.Flex

OCPU Count: 2

Network Bandwidth (Gbps): 2

Memory (GB): 32

Local Disk: Block Storage Only

Instance Access

The instance requires a public IP address to connect to it from the internet. You can assign either an ephemeral public IP or a reserved public IP to a private IP for this purpose.

Public IP Address: -

Username: opc

Primary VNIC

Private IP Address: 10.0.0.3

Network Security Groups: None Edit

Internal FQDN: ujanke-asset-private-vm... Show Copy

Subnet: Public Subnet

Launch Options

NIC Attachment Type: VFIO

Remote Data Volume: PARAVIRTUALIZED

Firmware: UEFI_64

Boot Volume Type: PARAVIRTUALIZED

In-transit Encryption: Disabled

Metrics

Metrics are measurements that let you monitor the health, capacity, and performance of your resources.

Metric Namespace: oci_computeagent

Start time: Jan 5, 2021 11:59:26 AM

End time: Jan 5, 2021 12:59:26 PM

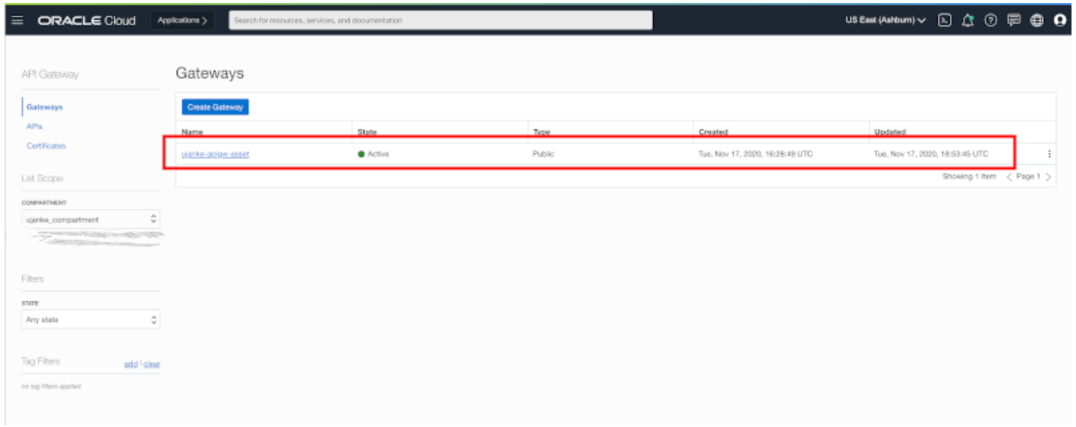
Quick Selects: Last hour

No Public IP address for the Private VM

Summarizing the changed architecture of the Public Cloud configuration as explained above, it turns out that we are reusing all components as introduced in previous blog. Under bottom line the main change is the introduction of a bastion server and the joint use of the underlying VCN.

Configuration of an API Gateway Deployment

As a next configuration step we will expose the unsecure REST service as secure API's via API Gateway. For this setup we have to start with a new Gateway configuration as shown in screenshot below.



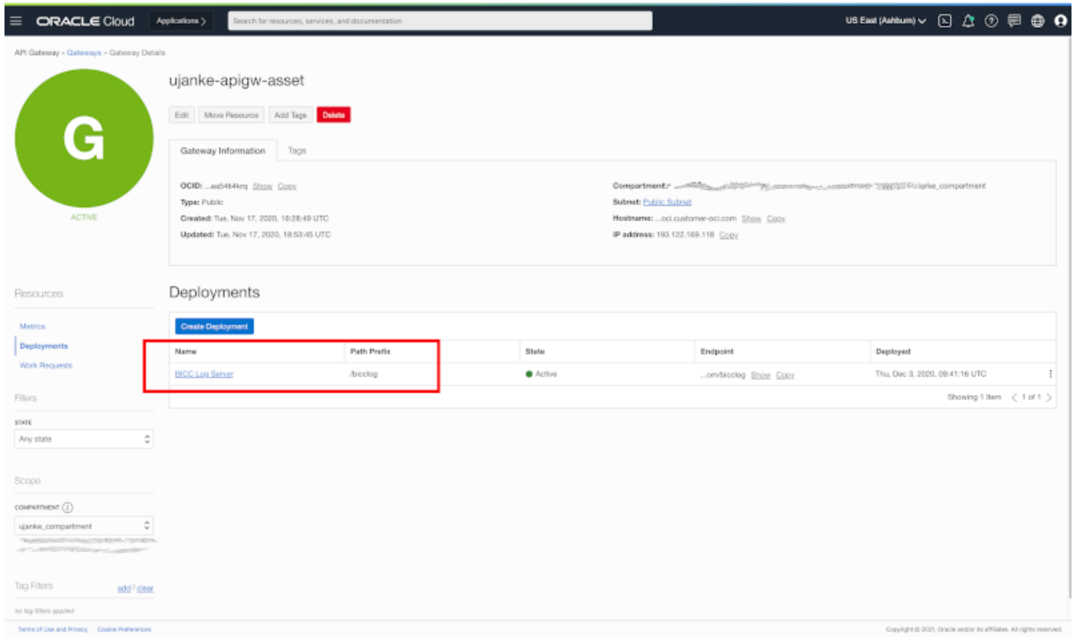
[redacted]

Name	State	Type	Created	Updated
ujanke-apigw-asset	● Active	Public	Tue, Nov 17, 2020, 16:28:49 UTC	Tue, Nov 17, 2020, 18:53:45 UTC

Showing 1 Item < Page 1 >

Creating an API Gateway Configuration for the BICC Logs REST API's

Once done we have to created an API Gateway deployment as a next step. The deployment will do the mappig for the internal REST API's and every single API meant to be published must be registered there.



ujanke-apigw-asset

OCID: ...ea54t4krq Show Copy
Type: Public
Created: Tue, Nov 17, 2020, 16:28:49 UTC
Updated: Tue, Nov 17, 2020, 18:53:45 UTC
Compartment: [redacted]ujanke_compartment
Subnet: Public Subnet
Hostname: ...oci.customer-oci.com Show Copy
IP address: 193.122.169.118 Copy

Deployments

Name	Path Prefix	State	Endpoint	Deployed
BICC Log Server	/biccIog	● Active	...om/bicclog Show Copy	Thu, Dec 3, 2020, 09:41:16 UTC

Showing 1 Item < 1 of 1 >

STATE: Any state

Scope

COMPARTMENT: ujanke_compartment
[redacted]

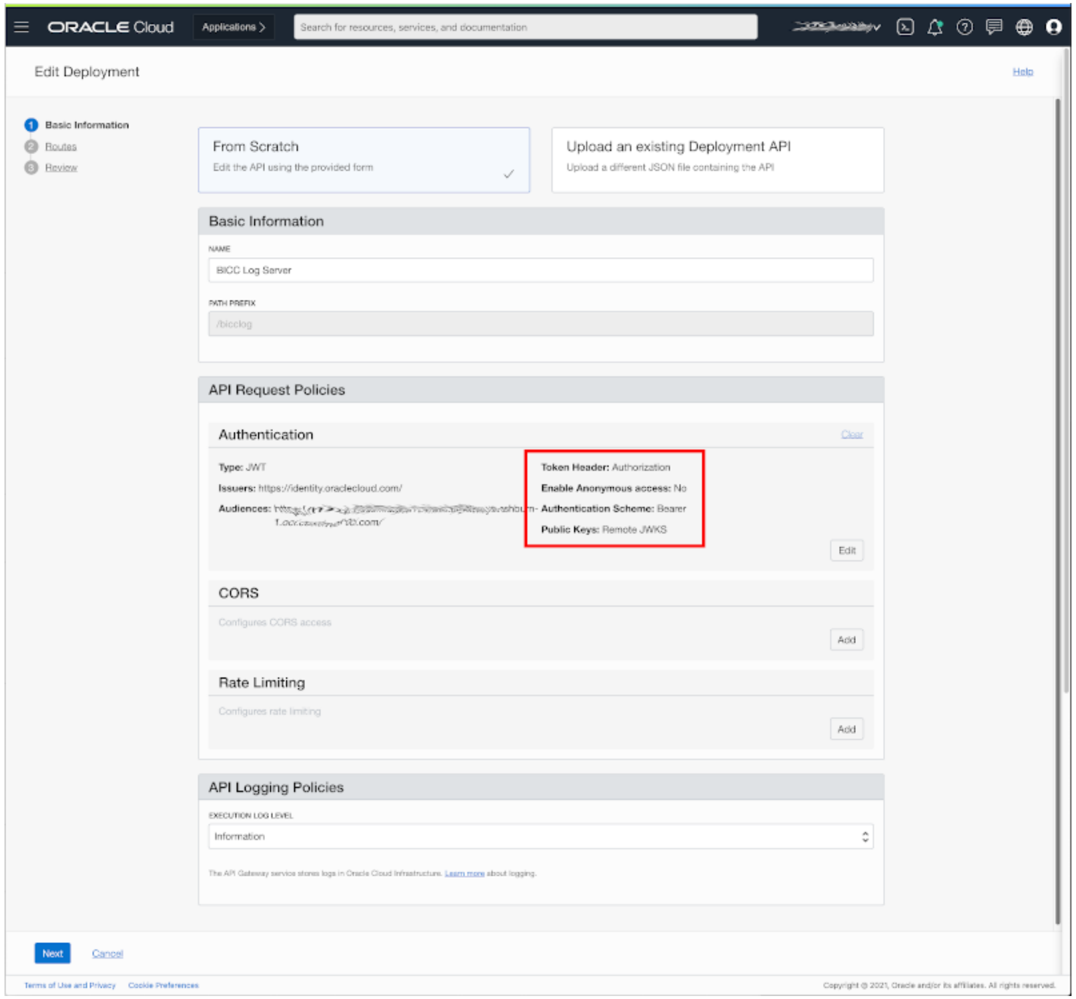
Tag Filters add | clear
no tag filters applied

Creating an API Gateway Configuration for the BICC Logs REST API's

The following multi-page configuration starts with the gathering of deployment level. The input of the API Request Policies. As shown below the following settings were made:

- Token Header requires an Authorization
- Anonymous access is denied - an invocation of any of the published REST API's requires a valid authentication via IDCS
- The Bearer Token is of type JWT and will be obtained from IDCS - as issuer we have to use the URL ***https://identity.oraclecloud.com/***
- As the audience we have to configure the Custom App registration URL in IDCS for this deployment

The activity mentioned in last bullet point above creates a challenge for our configuration: to enter this URL we must have this deployment being already registered in IDCS. But that is only possible after the configuration of this API Gateway deployment has been finished. To overcome this issue it is possible to leave the Authentication section open at the beginning and to finish the deployment configuration first. Once done we will configure the custom app registration in IDCS as a next step as explained further down below in this article.



Edit Deployment

1. Basic Information
2. Routes
3. Review

From Scratch ✓
Edit the API using the provided form

Upload an existing Deployment API
Upload a different JSON file containing the API

Basic Information

NAME: BICC Log Server

PATH PREFIX: /bicclog

API Request Policies

Authentication

Type: JWT
Issuers: https://identity.oraclecloud.com/
Audiences: [redacted]

Token Header: Authorization
Enable Anonymous access: No
Authentication Scheme: Bearer
Public Keys: Remote JWKS

CORS

Configures CORS access

Rate Limiting

Configures rate limiting

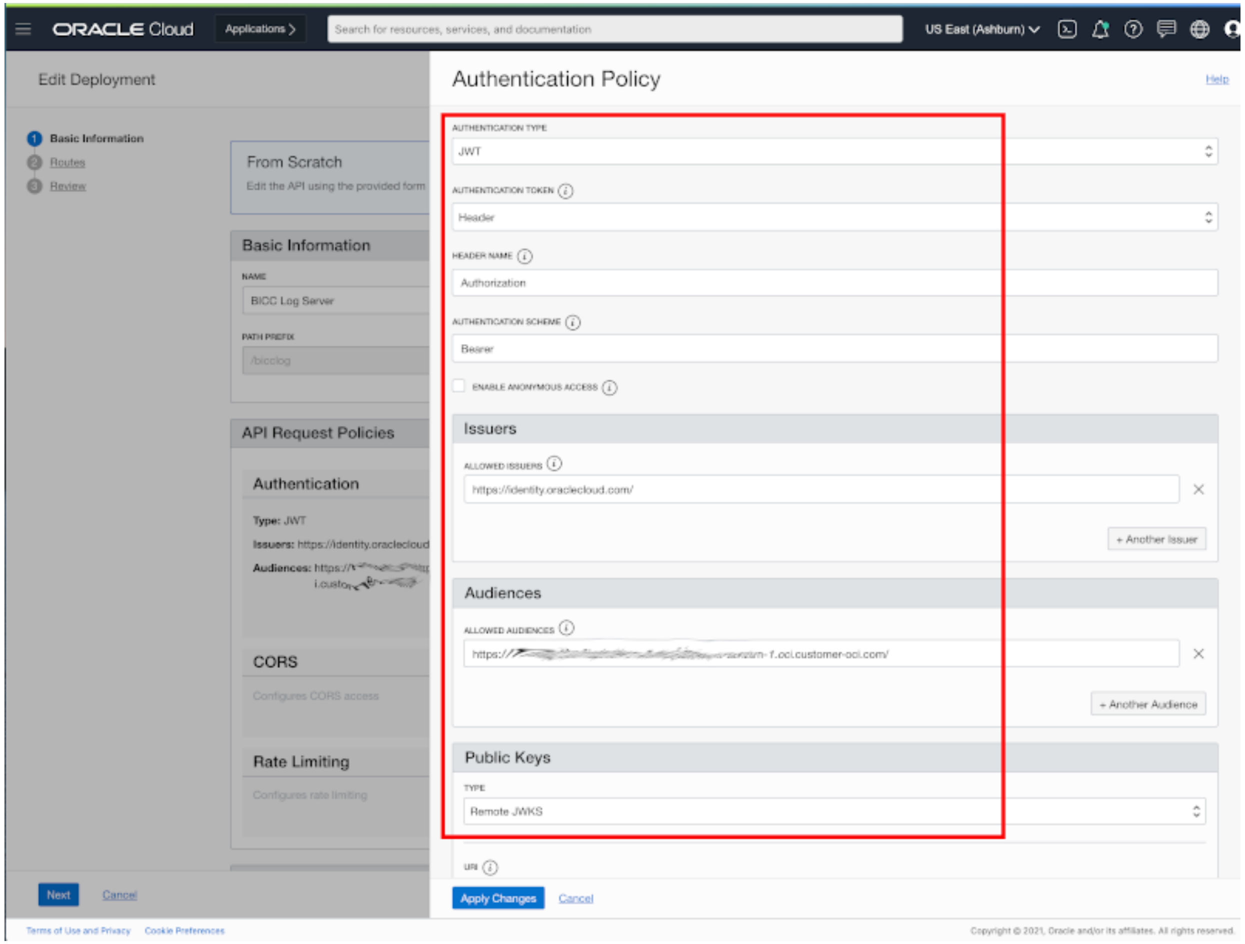
API Logging Policies

EXECUTION LOG LEVEL: Information

The API Gateway service stores logs in Oracle Cloud Infrastructure. Learn more about logging.

Configure the Authentication Policy to JWT using Bearer Tokens

The screenshot below shows the configuration page to enter the Authentication Policy values.



ORACLE Cloud
Applications
Search for resources, services, and documentation
US East (Ashburn)
Help
Terms of Use and Privacy
Cookie Preferences
Copyright © 2021, Oracle and/or its affiliates. All rights reserved.

Edit Deployment

1. Basic Information
2. Routes
3. Review

From Scratch
Edit the API using the provided form

Basic Information

NAME: BICC Log Server

PATH PREFIX: /bicclog

API Request Policies

Authentication

Type: JWT

Issuers: https://identity.oraclecloud

Audiences: https://[redacted] i.custo[redacted]

CORS

Configures CORS access

Rate Limiting

Configures rate limiting

Next Cancel

Authentication Policy

AUTHENTICATION TYPE: JWT

AUTHENTICATION TOKEN: Header

HEADER NAME: Authorization

AUTHENTICATION SCHEME: Bearer

□ ENABLE ANONYMOUS ACCESS

Issuers

ALLOWED ISSUERS: https://identity.oraclecloud.com/

+ Another Issuer

Audiences

ALLOWED AUDIENCES: https://[redacted]-f.oci.customer-oci.com/

+ Another Audience

Public Keys

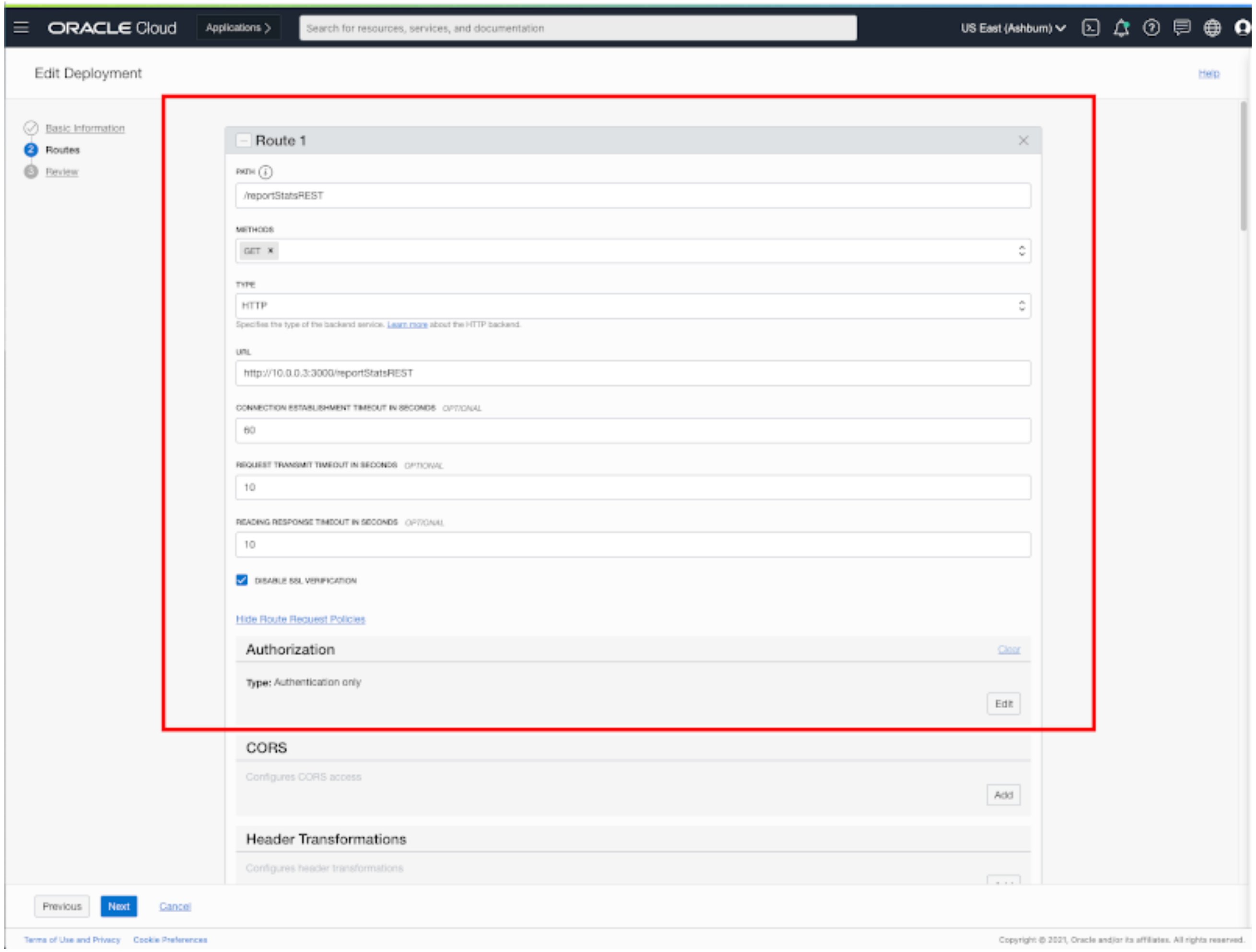
TYPE: Remote JWKS

URI

Apply Changes Cancel

Details of Authentication Policy configuration to to use JWT Bearer Tokens

As a next configuration step the single REST API's will be registered in API Gateway - called ***Routes***. API Gateway and the REST API's belong to the same VCN and therefore the internal/private IP addresses can be reached via this route and will be published as a public service.



Route 1

PATH: /reportStatsREST

METHODS: GET

TYPE: HTTP

Specifies the type of the backend service. Learn more about the HTTP backend.

URL: http://10.0.0.3:3000/reportStatsREST

CONNECTION ESTABLISHMENT TIMEOUT IN SECONDS *OPTIONAL*: 60

REQUEST TRANSMIT TIMEOUT IN SECONDS *OPTIONAL*: 10

READING RESPONSE TIMEOUT IN SECONDS *OPTIONAL*: 10

☑ DISABLE SSL VERIFICATION

Authorization

Type: Authentication only

CORS

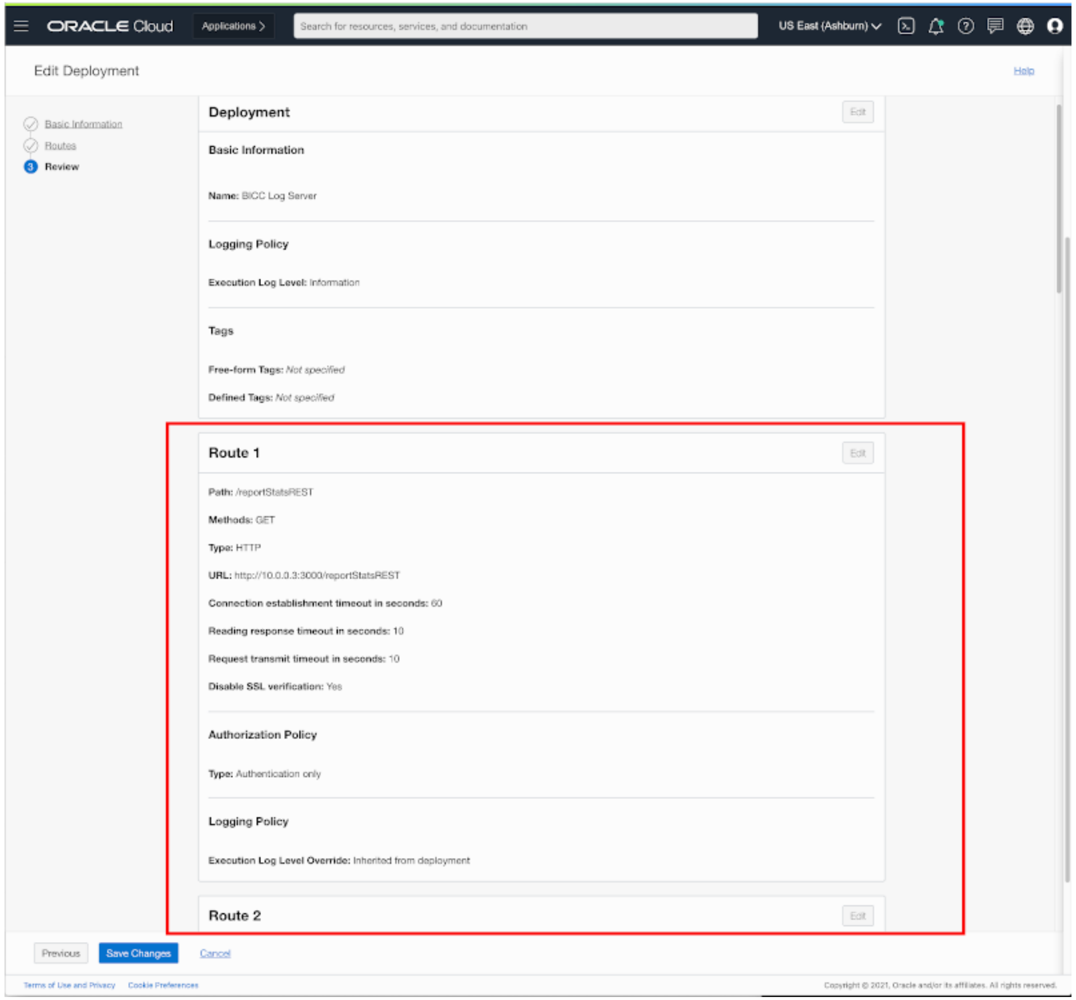
Configures CORS access

Header Transformations

Configures header transformations

Creating a Route for each published REST API

Every single route can be configured individually in terms of using a sprecific method, timeouts etc.



Edit Deployment

Deployment

Basic Information

Name: BICC Log Server

Logging Policy

Execution Log Level: Information

Tags

Free-form Tags: *Not specified*

Defined Tags: *Not specified*

Route 1

Path: /reportStatsREST

Methods: GET

Type: HTTP

URL: http://10.0.0.3:3000/reportStatsREST

Connection establishment timeout in seconds: 60

Reading response timeout in seconds: 10

Request transmit timeout in seconds: 10

Disable SSL verification: Yes

Authorization Policy

Type: Authentication only

Logging Policy

Execution Log Level Override: Inherited from deployment

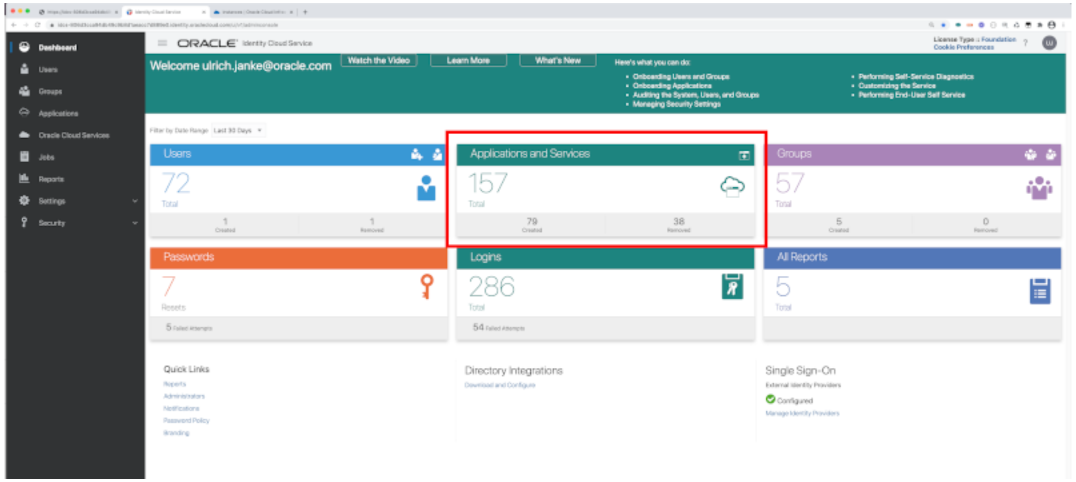
Route 2

`Route details showing that this REST API is private and secured`

Once these routes are fully oonfigured, we're ready to enter the the registration of a custom app in IDCS. That will ensure only authorized users or groups will be able to invoke the configured REST API's.

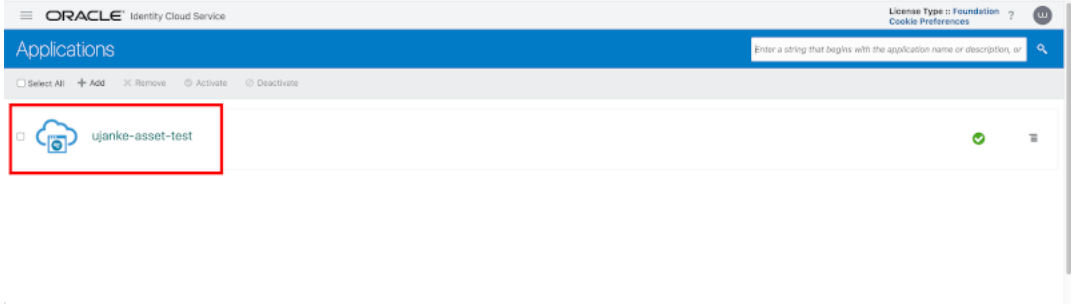
Setup a Custom App in IDCS

Registering a custom application in IDCS is a very sensitive activity that requires Administrator rights. In the following explanation we will see how such a registration can be performed for the API Gateway deployment we've done before. The IDCS Dashboard has a component showing all existent Applications and Services that will include also custom applications.



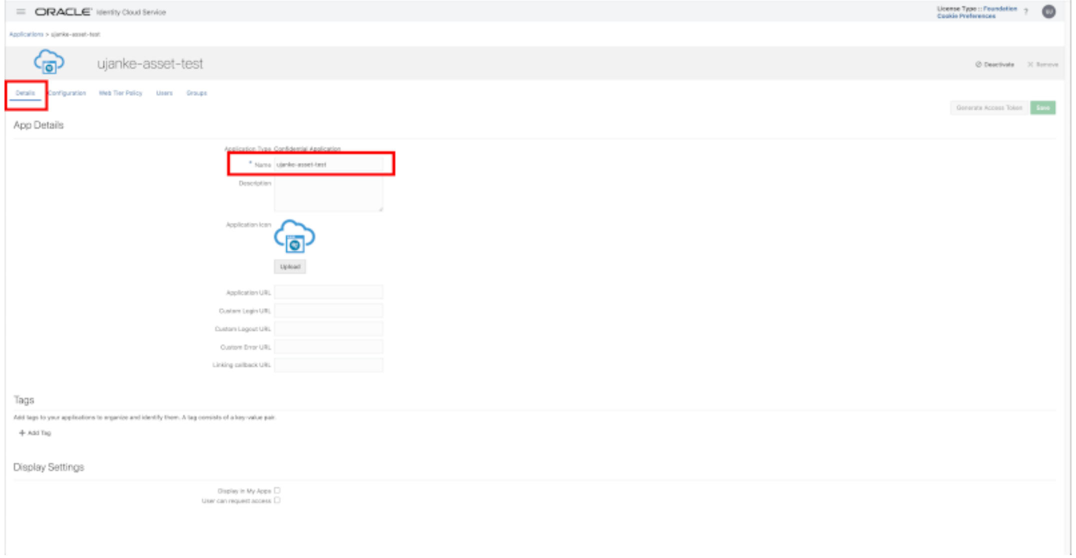
```
Start with the IDCS Dashboard
```

After finishing the setup for our registration of the API Gateway deployment it will be visibla, searchable and editable like all other apps and services. The screenshot below shows the target situation we're going to achievw with the following setup steps.


ORACLE Identity Cloud Service
License Type :: Foundation
Cookie Preferences
Applications
Enter a string that begins with the application name or description, or
Select All
Add
Remove
Activate
Deactivate
ujanke-asset-test

Showing a Custom App registration once finished

The configuration starts with a new application creation operation that will open up a new page as shown below. Some of the configuration activities can skipped while others are essential. We'll explain the steps that have to be done in order to have the application registered successfully. No further configuration steps are required in the Details page than entering the Application Name as shown below.



Application Type Confidential Application

* Name ujanke-asset-test

Description

Application Icon

Application URL

Custom Login URL

Custom Logout URL

Custom Error URL

Linking callback URL

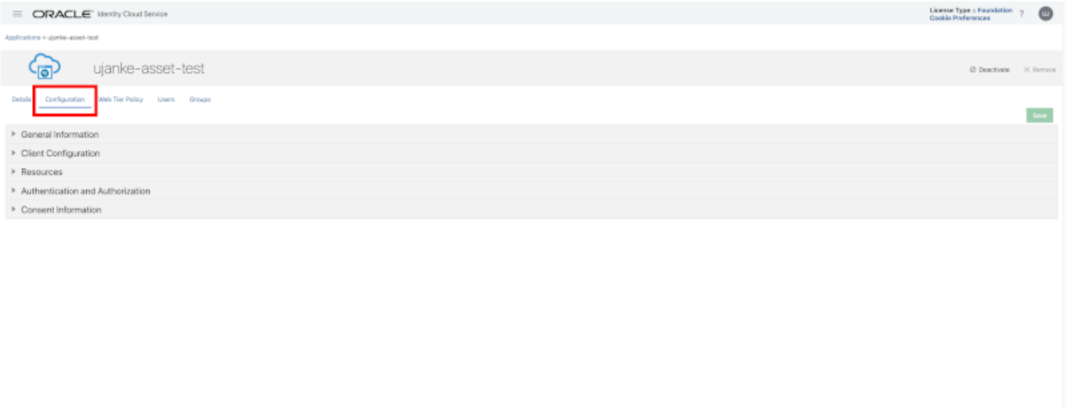
Add tags to your applications to organize and identify them. A tag consists of a key-value pair.

Display in My Apps □

User can request access □

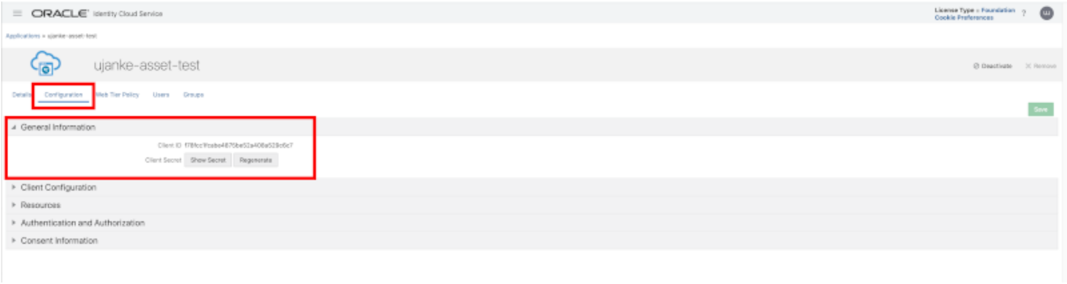
Custom Application registration – Details

The Configuration page below will be used to enter the most setup data for the registration of resources, authentication and authorization information. Further down below the single configuration steps will be explained in detail.



Custom Application registration – Configuration

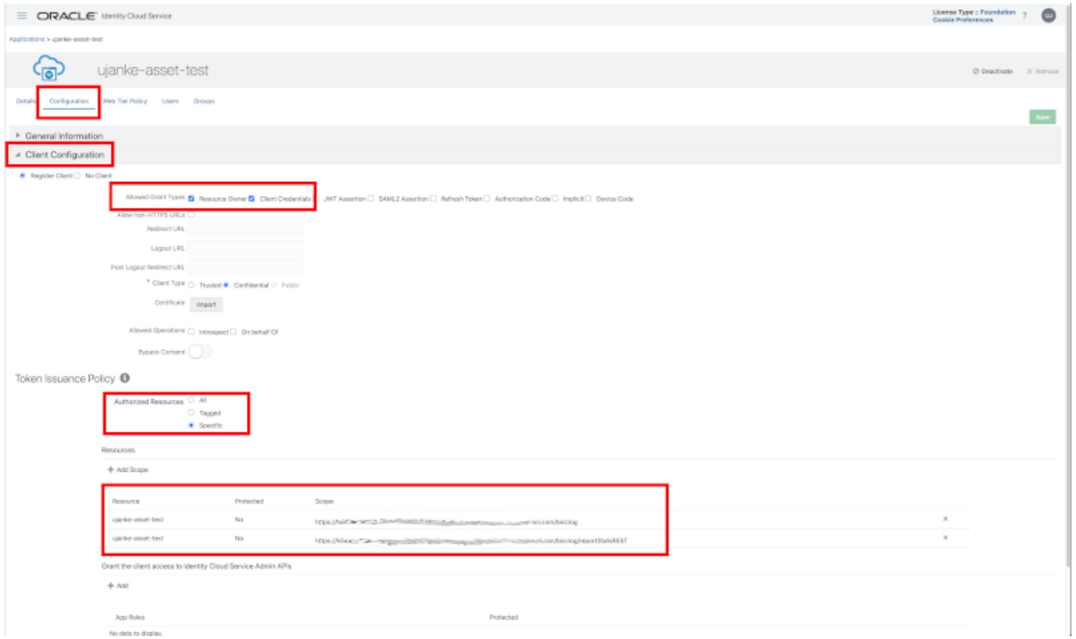
In the section General Information we can find the information for the Client ID and Client Secret that will be used to obtain the Bearer Token from IDCS. The Client Secret can ge regenerated in this page and the usage to obtain a Bearer Token will be shown in the next section using a sample in Postman.


Client ID f78fcc1fcabe4876be52a408a529c6c7

Custom Application registration – General Information

In section Client Configuration we configured the following values:

- Choose ***Register Client*** to enable the registration of of the services belonging to the API Gateway deployment
- Allowed Grant Types chosen for our configuration are ***Resource Owner*** and ***Client Credentials***
- Under ***Token Issuance Policy*** the value ***Specific*** must be chosen as an entry for ***Authorized Resources***
- Finally the API deployment resources must be added as a ***Scope*** - as shown via a registration sample below the root path (here *https://<API GW Deployment>/bicclog*) must be registered, but also every single API like *https://<API GW Deployment>/bicclog/reportStatsREST*



● Register Client ○ No Client

Allowed Grant Types ☑ Resource Owner ☑ Client Credentials □ JWT Assertion □ SAML2 Assertion □ Refresh Token □ Authorization Code □ Implicit □ Device Code

Allow non-HTTPS URLs □

Redirect URL

Logout URL

Post Logout Redirect URL

* Client Type ○ Trusted ● Confidential ○ Public

Certificate Import

Allowed Operations □ Introspect □ On behalf Of

Bypass Consent

Token Issuance Policy

Authorized Resources ○ All ○ Tagged ● Specific

Resources

+ Add Scope

Resource	Protected	Scope
ujanke-asset-test	No	https://[redacted]/bicclog
ujanke-asset-test	No	https://[redacted]/bicclog/reportStatsREST

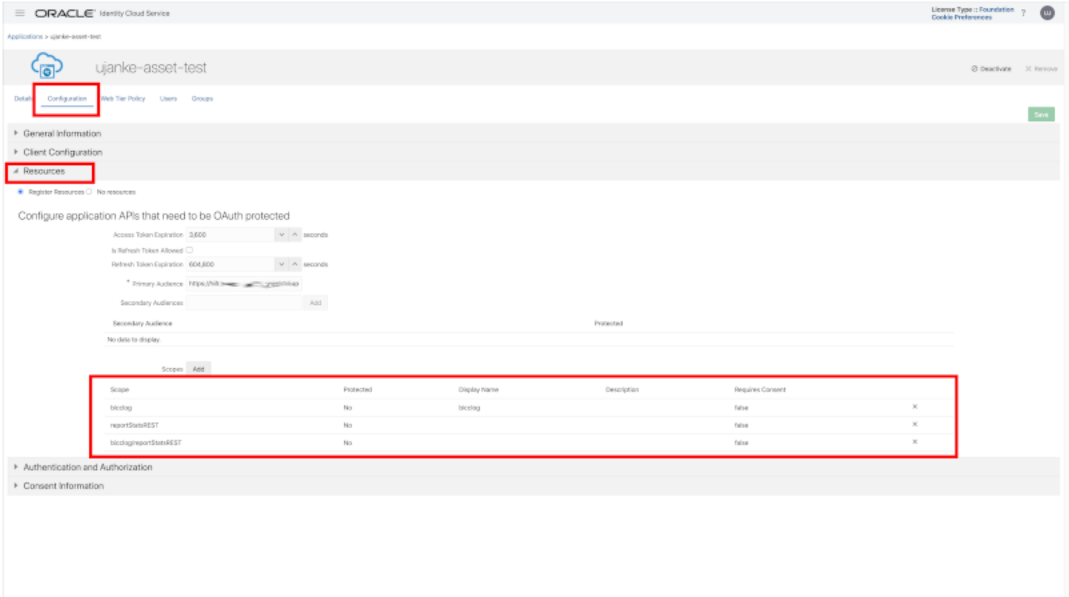
Grant the client access to Identity Cloud Service Admin APIs

+ Add

App Roles	Protected
No data to display.	

Custom Application registration – Client Configuration

Similar as shown above, we have to register every single API under the ***Resources*** section. Ihere we will add every single REST API as configured in the API Gateway deployment to the list of configured API's that are protected by OAuth. For this purpose we will register resources via ***Add Scope*** to this group of API's. Important for this configuration: via the mandatory field ***Primary Audience*** the common URL for all of these API's must be entered, while the scope entries below are relativ to this URL. Other commonly used parameters are the token related expiration and refresh timings.



● Register Resources ○ No resources

Configure application APIs that need to be OAuth protected

Access Token Expiration 3,600 seconds

Is Refresh Token Allowed □

Refresh Token Expiration 604,800 seconds

* Primary Audience https://[redacted]

Secondary Audiences Add

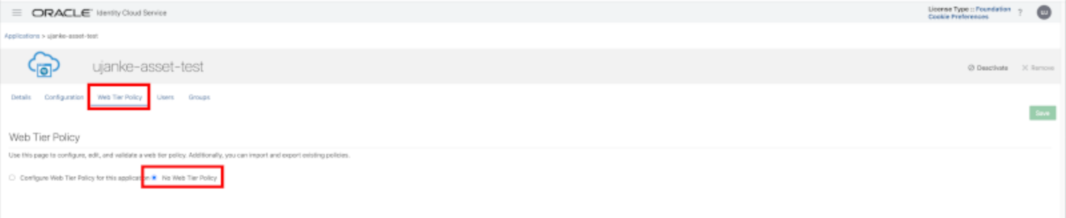
Secondary Audience	Protected
No data to display.	

Scopes Add

Scope	Protected	Display Name	Description	Requires Consent	
biccdog	No	biccdog		false	×
reportStatsREST	No			false	×
biccdog/reportStatsREST	No			false	×

Custom Application registration – Resources details configuration

As shown below we did not configure any specific web tier policies in this sample.



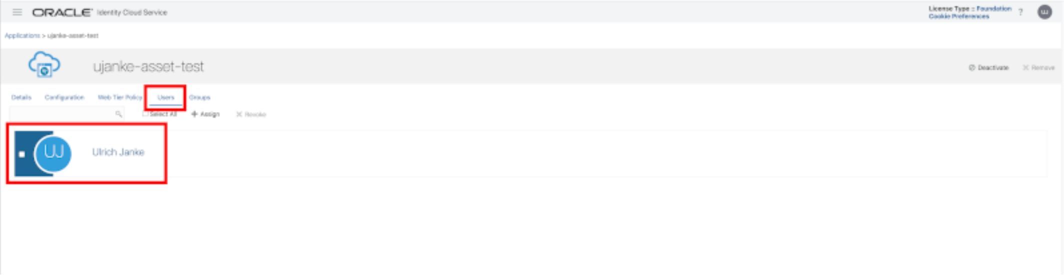
Web Tier Policy

Use this page to configure, edit, and validate a web tier policy. Additionally, you can import and export existing policies.

○ Configure Web Tier Policy for this application ● No Web Tier Policy

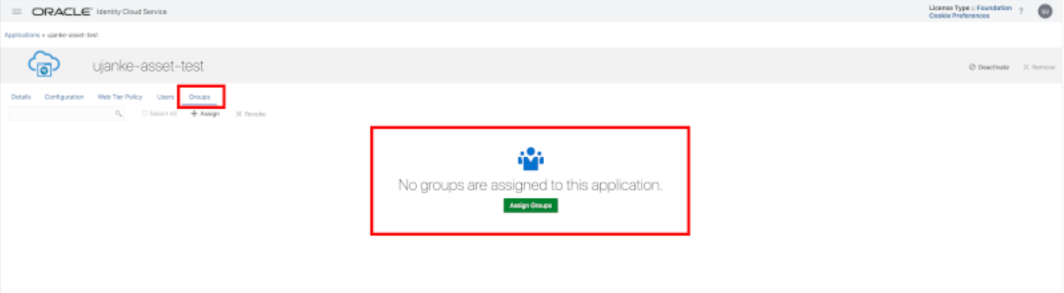
Custom Application registration – Web Tier Policy

At the end of this IDCS Custom App setup we must assign the configured resources to dedicated users and/or groups. That will grant access to these secured REST API's to a number of specific users.



Custom Application registration – Users

In this sample we assigned the permissions to just one user and left the group entry unset.



Custom Application registration – Groups

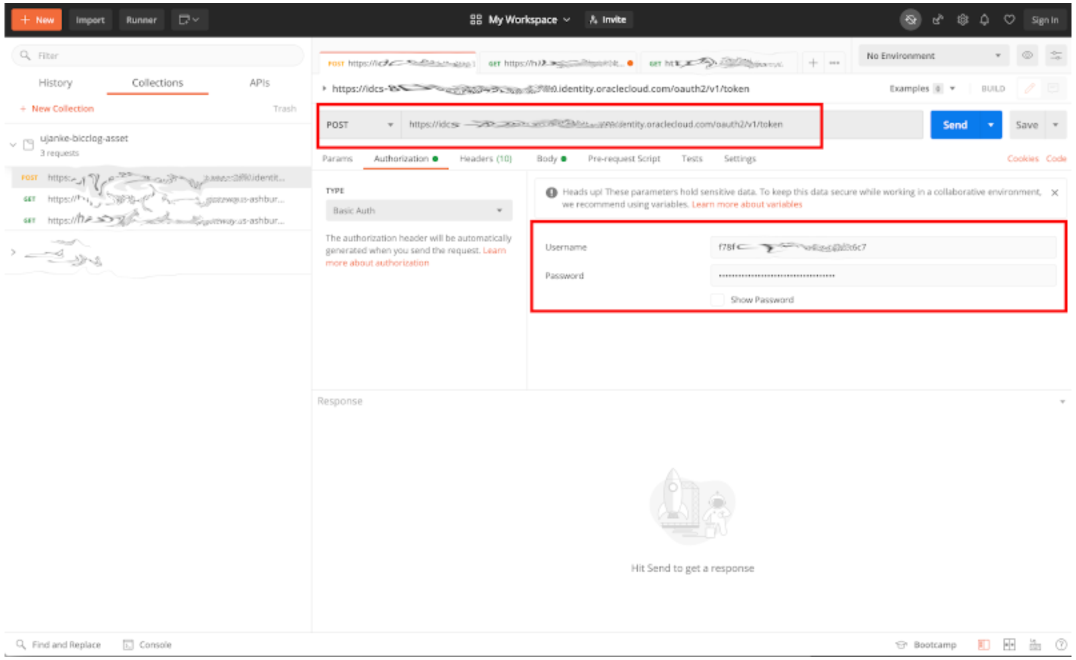
Now, after finishing the creation of a custom IDCS App, the REST API's can't be accessed anymore without an authenticaion and authorization. As mentioned above latest now its also possible to update the API Request Policies in API Gateway with the specific information in the ***Audiences*** field.

Testing the secure BICC Log REST Server API ...

By performing the security setup above we got a set pf protected API's that can only be used by users and groups being authorized via IDCS. As explained this was possible without any modification in the source code of the the application. From here the REST API's can be invoked from other cloud based services like ***Visual Builder*** or ***Oracle Integration Cloud***.
For a simple test of these protected API's we use ***Postman*** as shown below. It will demonstrate how to login to IDCS to obtain a bearer token before invoking a protected REST API.
This test scenario is two-fold:

1. Via the tennant-specific IDCS URL we're obtaining the bearer token
2. Using the token we can invoke the protected REST API

As shown in screenshot below we have to configure and run a POST call first by using the URL as it comes in the format *https://idcs-<ID>.identity.oraclecloud.com/ui/v1/signin* and can be obtained from the login page of this tenancy. When using Postman we have to provide a manual authentication method where we must enter a username and password that can be obtained from the IDCS configuration page for our custom application. The sample below shows how these values can be entered in the *Authorization* tab.



POST https://idc[redacted]

GET https://h[redacted]

GET ht[redacted]

https://idcs-[redacted]0.identity.oraclecloud.com/oauth2/v1/token

POST https://idcs-[redacted]identity.oraclecloud.com/oauth2/v1/token

POST https:[redacted]identit...

GET https://[redacted]gateway.us-ashbur...

GET https://[redacted]gateway.us-ashbur...

[redacted]

TYPE

Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Username f78f[redacted]c6c7

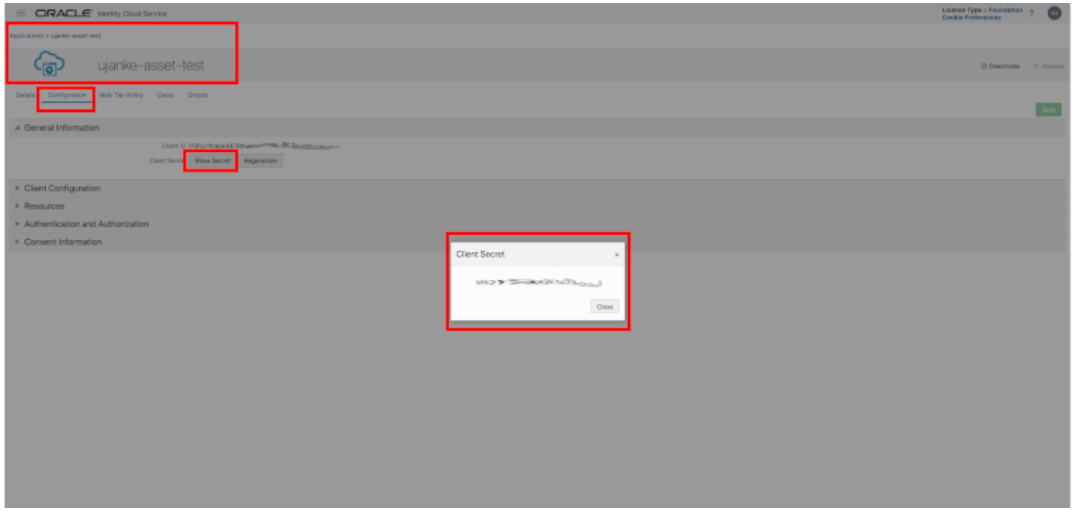
Password ••••••••••••••••••••••••••••••••••••

□ Show Password

Hit Send to get a response

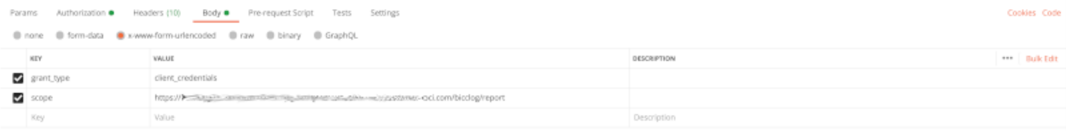
Testing via Postman – entering the IDCS credentials to obtain a token

The screenshot below shows how these values for username and password can be obtained in IDCS.



Testing via Postman – How to obtain the Client Secret in IDCS

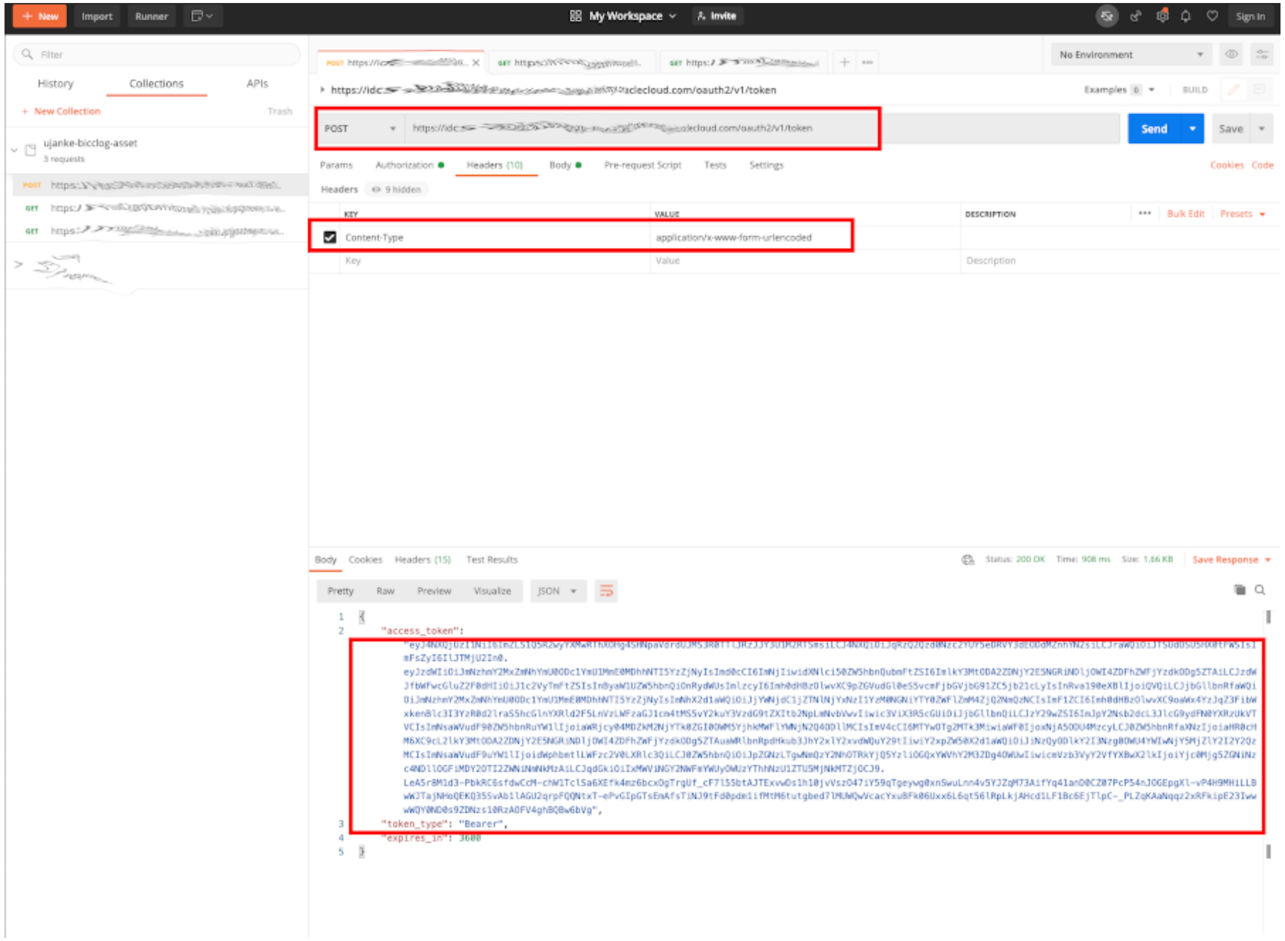
Before running the request we have to set an additional parameter: the *Body* must contain an entry *scope* that contains the resource for which we're obtaining the token. Obviously that must be an entry that has been configured in IDCS in the custom app setup. As shown in sample below we're using the URL of a protected REST API in API Gateway ending with *bicclog/reportStatsREST*.



	KEY	VALUE	DESCRIPTION
☑	grant_type	client_credentials	
☑	scope	https://[redacted]customer-oci.com/bicclog/report	
	Key	Value	Description

Testing via Postman – entering the scope credentials to obtain a token

A successful execution of this call will return a JSON structure that contains the value for an *access_token*. This is the token we need in our second step to make the API call. We must also ensure to use *application/x-www-form-urlencoded* as *Content-Type* in the request header.



POST https://idc[redacted]aclecloud.com/oauth2/v1/token

	KEY	VALUE	DESCRIPTION
☑	Content-Type	application/x-www-form-urlencoded	
	Key	Value	Description

Status: 200 OK Time: 908 ms Size: 1.66 KB

```
{
    "access_token":
        "eyJ4NXQjUzI1NiI6ImZLS1Q5R2wyYXMwRThXOHg4SHNpaVdrdUJMS3R0TTlJRzJJY3U1M2RTSmsiLCJ4NXQiOiJqRzQ2QzdONzc2YUY5eDRVY3dEODdMZnhYN2siLCJraWQiOiJTSUdOSU5HX0tFWSIsI
        mFsZyI6IlJTMjU2In0.
        eyJzdWIiOiJmNzhmY2MxZmNhYmU0ODc1YmU1MmE0MDhhNTI5YzZjNyIsImd0cCI6ImNjIiwidXNlci50ZW5hbnQubmFtZSI6ImlkY3MtODA2ZDNjY2E5NGRiNDljOWI4ZDFhZWFjYzdkODg5ZTAiLCJzdW
        JfbWFwcGluZ2F0dHIiOiJ1c2VyTmFtZSIsInByaW1UZW5hbnQiOnRydWUsImlzcyI6Imh0dHBzOlwvXC9pZGVudGl0eS5vcmFjbGVjbG91ZC5jb21cLyIsInRva190eXBlIjoiQVQiLCJjbGllbnRfaWQi
        OiJmNzhmY2MxZmNhYmU0ODc1YmU1MmE0MDhhNTI5YzZjNyIsImNhX2d1aWQiOiJjYWNjdC1jZTNlNjYxNzI1YzM0NGNiYTY0ZWFlZmM4ZjQ2NmQzNCIsImF1ZCI6Imh0dHBzOlwvXC9oaWx4YzJqZ3FibW
        xkenBlc3I3YzR0d2lraS5hcGlnYXRld2F5LnVzLWFzaGJ1cm4tMS5vY2kuY3VzdG9tZXItb2NpLmNvbVwvIiwic3ViX3R5cGUiOiJjbGllbnQiLCJzY29wZSI6ImJpY2Nsb2dcL3JlcG9ydFN0YXRzUkVT
        VCIsImNsaWVudF90ZW5hbnRuYW1lIjoiaWRjcy04MDZkM2NjYTk0ZGI0OWM5YjhkMWFlYWNjN2Q4ODllMCIsImV4cCI6MTYwOTg2MTk3MiwiaWF0IjoxNjA5ODU4MzcyLCJ0ZW5hbnRfaXNzIjoiaHR0cH
        M6XC9cL2lkY3MtODA2ZDNjY2E5NGRiNDljOWI4ZDFhZWFjYzdkODg5ZTAuaWRlbnRpdHkub3JhY2xlY2xvdWQuY29tIiwiY2xpZW50X2d1aWQiOiJiNzQyODlkY2I3Nzg0OWU4YWIwNjY5MjZlY2I2Y2Qz
        MCIsImNsaWVudF9uYW1lIjoidWphbmtlLWFzc2V0LXRlc3QiLCJ0ZW5hbnQiOiJpZGNzLTgwNmQzY2NhOTRkYjQ5YzliOGQxYWVhY2M3ZDg4OWUwIiwicmVzb3VyY2VfYXBwX2lkIjoiYjc0Mjg5ZGNiNz
        c4NDllOGFiMDY2OTI2ZWNiNmNkMzAiLCJqdGkiOiIxMWViNGY2NWFmYWUyOWUzYThhNzU1ZTU5MjNkMTZjOCJ9.
        LeA5r8M1d3-PbkRC6sfdwCcM-chW1TclSa6XEfk4mz6bcxOgTrqUf_cF7l55btAJTExvwDs1h10jvVsz047iY59qTgeywg0xnSwuLnn4v5YJZqM73AifYq41anD0CZ07PcP54nJOGEpgXl-vP4H9MHiLLB
        wWJTajNHoQEKQ35SvAb1lAGU2qrpFQQNtxT-ePvGIpGTsEmAfsTiNJ9tFd0pdm1ifMtM6tutgbed7lMUWQwVcacYxuBFk06Uxx6L6qt56lRpLkjAHcd1LF18c6EjTlpC-_PLZqKAaNqqz2xRFkipE23Iww
        wWQY0ND0s9ZDNzs10RzA0FV4ghBQBw6bVg",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

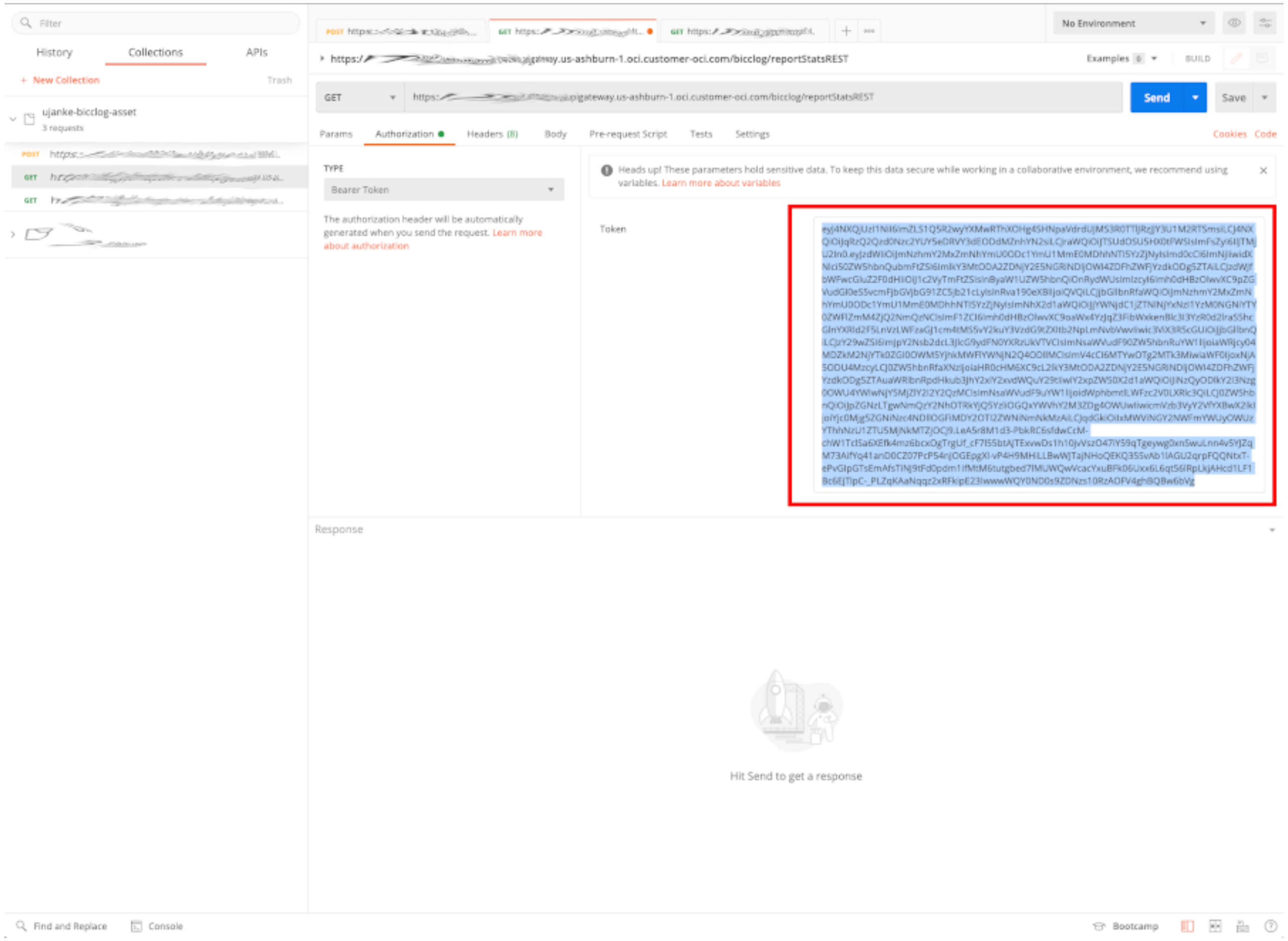
Testing via Postman – Receiving the Bearer Token

An alternative way to obtain this *access_token* would be a *curl* command execution as shown below. It uses the same autorization values and other information as shown above in Postman. Instead of passing a basic authentication via client user id and passwort we are providing a base64 encoded entry for these both values.

```
        curl --location --request POST 'https://idcs-
806Rgh78020IkjKi363484740mshkfjs79fd89e0.identity.oraclecloud.com/oauth2/v1/token' \
        --header 'Content-Type: application/x-www-form-urlencoded' \
        --header 'Authorization: Basic
Zjc4ZmNjMW89nHopvZjYWJlNDg3NWJlNTWM2Yzc6Yjg1MmY5YjAtYzk3Ni00MGY2LWFjZjgtNjEzYWQ2MTMzOTA1' \
        --data-urlencode 'grant_type=client_credentials' \
        --data-urlencode 'scope=https://fghuflen836d7e9c87fhj33j4i.apigateway.us-washturn-8.oci.customer-
oci.com/bicclog/reportStatsREST'
```

```
        {"access_token":"eyJ4NXQjUzI1NiI6ImZLS1Q5R2wyYXMwRThXOHg4SHNpaVdrdUJMS3R0TTlJRzJJY3U1M2RTSmsiLCJ4NXQiOiJqRzQ
2Qzd0Nzc2YUY5eDRVY3dEODdMZnhYN2siLCJraWQiOiJTSUdOSU5HX0tFWSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiJmNzhmY2MxZmNhYmU0ODc1YmU
1MmE0MDhhNTI5YzZjNyIsImd0cCI6ImNjIiwidXNlci50ZW5hbnQubmFtZSI6ImlkY3MtODA2ZDNjY2E5NGRiNDljOWI4ZDFhZWFjYzdkODg5ZTAiLCJ
zdWJfbWFwcGluZ2F0dHIiOiJ1c2VyTmFtZSIsInByaW1UZW5hbnQiOnRydWUsImlzcyI6Imh0dHBzOlwvXC9pZGVudGl0eS5vcmFjbGVjbG91ZC5jb21
cLyIsInRva190eXBlIjoiQVQiLCJjbGllbnRfaWQiOiJmNzhmY2MxZmNhYmU0ODc1YmU1MmE0MDhhNTI5YzZjNyIsImNhX2d1aWQiOiJjYWNjdC1jZTN
lNjYxNzI1YzM0NGNiYTY0ZWFlZmM4ZjQI6Imh0dHBzOlwvXC9oaWx4YzJqZ3FibWxkenBlc3I3YzR0d2lraS5hcGlnYXRld2F5LnVzLWFzaGJ1cm4tMS
5vY2kuY3VzdG9tZXItb2NpLmNvbVwvIiwic3ViX3R5cGUiOiJjbGllbnQiLCJzY29wZSI6ImJpY2Nsb2dcL3JlcG9ydFN0YXRzUkVTVCIsImNsaWVudF
90ZW5hbnRuYW1lIjoiaWRjcy04MDZkM2NjYTk0ZGI0OWM5YjhkMWFlYWNjN2Q4ODllMCIsImV4cCI6MTYxMzE0NzM0NCwiaWF0IjoxNjEzMTQzNzQ0LC
J0ZW5hbnRfaXNzIjoiaHR0cHM6XC9cL2lkY3MtODA2ZDNjY2E5NGRiNDljOWI4ZDFhZWFj9403f0pefhB679RlbnRpdHkub3JhY2xlY2xvdWQuY29tIi
wiY2xpZW50X2d1aWQiOiJiNzQyODlkY2I3Nzg0OWU4YWIwNjY5MjZlY2I2Y2QzMCIsImNsaWVudF9uYW1lIjoidWphbmtlLWFzc2V0LXRlc3QiLCJ0ZW
5hbnQiOiJpZGNzLTgwNmQzY2NhOTRkYjQ5YzliOGQxYWVhY2M3ZDg4OWUwIiwicmVzb3VyY2VfYXBwX2lkIjoiYjc0Mjg5ZGNiNzc4NDllOGFiMDY2OT
I2ZWNiNmNkMzAiLCJqdGkiOiIxMWViNmQ0NzA5ZGQ1ZDc4YWMyZjdmM2FhY2IxYjk1ZCJ9.tDuyQSDRoZsJ5N7cAusllfuVcmnEQMJ9QNTYIdaLaSnRp
VNSWRX3IsxTOkX266Y9AYjGhwND4Pqo86VZMnQOes3kLRqPpgEnXUym_bDGKVgZ13sc0sgAPcmvSLS1j4srAYKua8njL3hP82UIdqIJop3npFFoSapFY
3DqjYkLY8XijxXQgn76tHPRusmvRv1Gvy_Y_blbY3koCOh5oAVhyeYlPhuuyOWkEn2FX3CTYUTgpMCwfnZaiquHfBVraSOUlvg1k_6HEI42HiJNvIGvz
vNPLUokxN_pdS-BwoLaeb8XpZ_tbtSq8pZ30jI3l57hBt-GJKdu_hpKhaI36butug","token_type":"Bearer","expires_in":3600}
```

Once we retrieved the bearer token we can now perform a *GET* request to invoke the protected REST API. As shown below we have to chose *Bearer Token* as autorization type and to copy the *access_token* from previous call and must insert the value into the *token* field. Afterwards we are ready to perform the request.



https:/[redacted]gateway.us-ashburn-1.oci.customer-oci.com/bicclog/reportStatsREST

GET https:[redacted]pigateway.us-ashburn-1.oci.customer-oci.com/bicclog/reportStatsREST

TYPE
Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

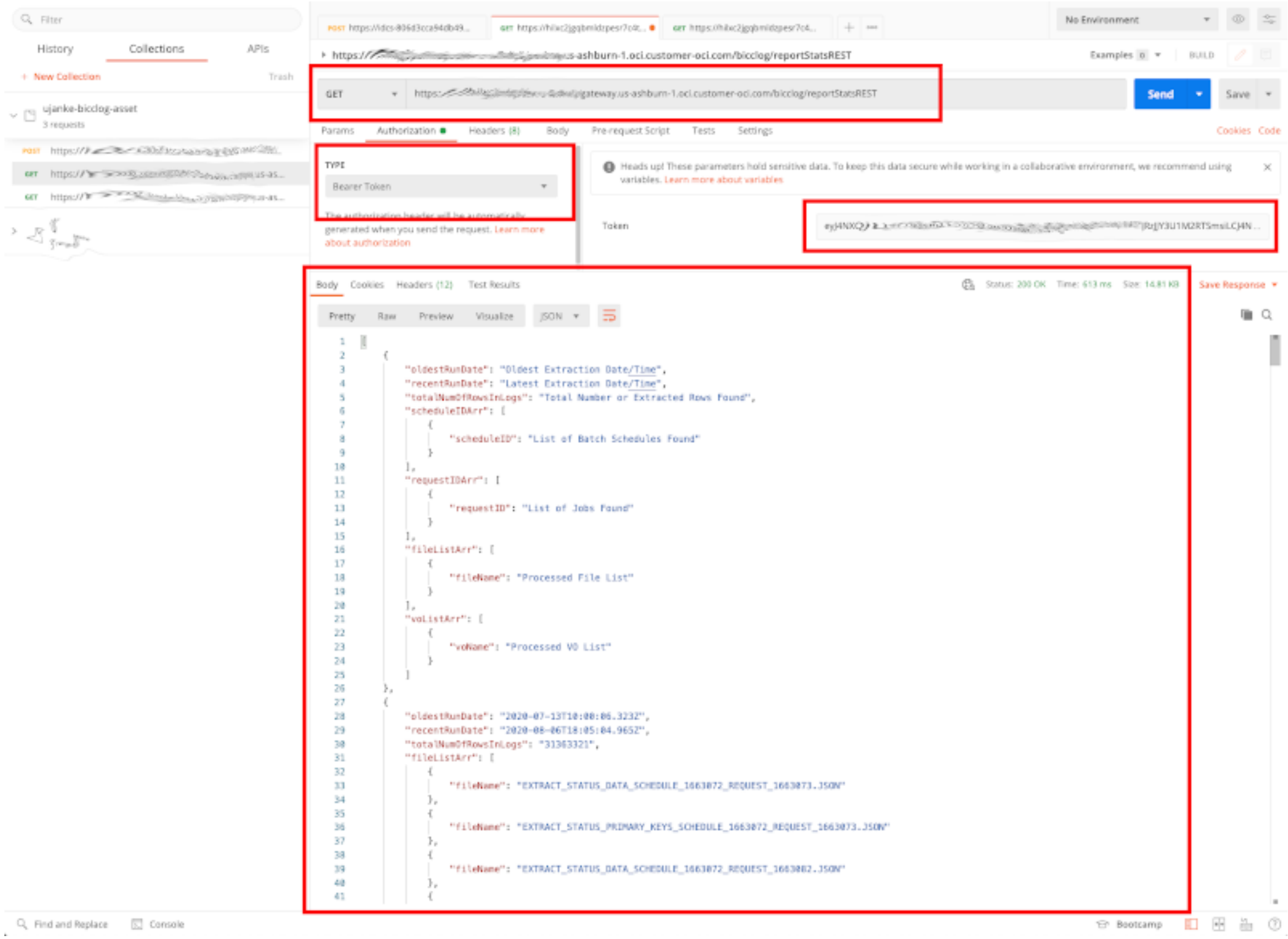
Token

eyJ4NXQjUzI1NiI6ImZLS1Q5R2wyYXMwRThXOHg4SHNpaVdrdUJMS3R0TTljRzJjY3U1M2RTSmsiLCJ4NXQiOiJqRzQ2Qzd0Nzc2YUY5eORVY3dEODdMZnhYN2siLCJraWQiOiJTSUdOSU5HX0tFWSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiJmNzhmY2MxZmNhYmU0ODc1YmU1MmE0MDhhNTI5YzZjNyIsImd0cCI6ImNjIiwidXNlci50ZW5hbnQubmFtZSI6ImlkY3MtODA2ZDNjY2E5NGRlNDljOWI4ZDFhZWFjYzdkODg5ZTAiLCJzdWJfbWFwcGluZ2F0dHIiOiJ1c2VyTmFtZSIsInByaW1UZW5hbnQiOnRydWUsImlzcyI6Imh0dHBzOlwvXC9pZGVudGl0eS5vcmFjbGVjbG91ZC5jb21cLyIsInRva190eXBlIjoiQVQiLCJjbGllbnRfaWQiOiJmNzhmY2MxZmNhYmU0ODc1YmU1MmE0MDhhNTI5YzZjNyIsImNhX2d1aWQiOiJjYWNjdC1jZTNlNjYxNzI1YzM0NGNiYTY0ZWFlZmM4ZjQ2NmQzNCIsImF1ZCI6Imh0dHBzOlwvXC9oaWx4YzJqZ3FibWxkenBlc3l3YzR0d2lraS5hcGlnYXRld2F5LnVzLWFzaGJ1cm4tMS5vY2kuY3VzdG9tZXItb2NpLmNvbVwvIiwic3ViX3R5cGUiOiJjbGllbnQiLCJzY29wZSI6ImJpY2Nsb2dcL3JlcG9ydFN0YXRzUkVTVCIsImNsaWVudF90ZW5hbnRuYW1lIjoiaWRjcy04MDZkM2NjYTk0ZGI0OWM5YjhkMWFlYWNjN2Q4ODllMCIsImV4cCI6MTYwOTg2MTk3MiwiaWF0IjoxNjA5ODU4MzcyLCJ0ZW5hbnRfaXNzIjoiaHR0cHM6XC9cL2lkY3MtODA2ZDNjY2E5NGRlNDljOWI4ZDFhZWFjYzdkODg5ZTAuaWRlbnRpdHkub3JhY2xlY2xvdWQuY29tIiwiY2xpZW50X2d1aWQiOiJiNzQyODlkY2I3Nzg0OWU4YWIwNjY5MjZlY2I2Y2QzMCIsImNsaWVudF9uYW1lIjoidWphbmtlLWFzc2V0LXRlc3QiLCJ0ZW5hbnQiOiJpZGNzLTgwNmQzY2NhOTRkYjQ5YzliOGQxYWVhY2M3ZDg4OWUwIiwicmVzb3VyY2VfYXBwX2lkIjoiYjc0Mjg5ZGNiNzc4NDllOGFiMDY2OTI2ZWNiNmNkMzAiLCJqdGkiOiIxMWVlNGY2NWFmYWUyOWUzYThhNzU1ZTU5MjNkMTZjOCJ9.LeA5r8M1d3-PbkRC6sfdwCcM-chW1TcI5a6XEfk4mz6bcxOgTrgUf_cF7I5SbtAjTExwwDs1h10jvVszO47iY59qTgeywg0xn5wuLnn4v5YJZqM73AifYq41anD0CZ07PcP54njOGEpgXl-vP4H9MHLLBwWJTajNHoQEKQ35SvAb1lAGU2qrpFQQNtxT-ePvGlpGTsEmAfsTlNJ9tFd0pdm1IfMtM6tutgbed7lMUWQwVcacYxuBFk06Uxx6L6qt56lRpLkjAHcd1LF1Bc6EjTlpC-_PLZqKAaNqqz2xRFkipE23lwwwWQY0ND0s9ZDNzs10RzAOFV4ghBQBw6bVg

Hit Send to get a response

Testing via Postman – Using the Bearer Token in REST API Call

Like shown below a successful invocation will result obtain the output as expected.



https://[redacted]s-ashburn-1.oci.customer-oci.com/bicclog/reportStatsREST

GET https://[redacted]gateway.us-ashburn-1.oci.customer-oci.com/bicclog/reportStatsREST

TYPE: Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Token: eyJ4NXQ[redacted]RzJY3U1M2RTSmsiLCJ4N...

Status: 200 OK Time: 613 ms Size: 14.81 KB

```
[
    {
        "oldestRunDate": "Oldest Extraction Date/Time",
        "recentRunDate": "Latest Extraction Date/Time",
        "totalNumOfRowsInLogs": "Total Number or Extracted Rows Found",
        "scheduleIDArr": [
            {
                "scheduleID": "List of Batch Schedules Found"
            }
        ],
        "requestIDArr": [
            {
                "requestID": "List of Jobs Found"
            }
        ],
        "fileListArr": [
            {
                "fileName": "Processed File List"
            }
        ],
        "voListArr": [
            {
                "voName": "Processed VO List"
            }
        ]
    },
    {
        "oldestRunDate": "2020-07-13T10:00:06.323Z",
        "recentRunDate": "2020-08-06T18:05:04.965Z",
        "totalNumOfRowsInLogs": "31363321",
        "fileListArr": [
            {
                "fileName": "EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663073.JSON"
            },
            {
                "fileName": "EXTRACT_STATUS_PRIMARY_KEYS_SCHEDULE_1663072_REQUEST_1663073.JSON"
            },
            {
                "fileName": "EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663082.JSON"
            },
            {
```

Testing via Postman – Results of a successful REST API Call

Same as shown above also here we can perform an alternate run on the comand line.

```
   curl --location --request GET 'https://hfjekendiele8jr9322jeh2k4i.apigateway.us-faraway-4.oci.customer-
oci.com/bicclog/reportStatsREST' \
     --header 'Content-Type: application/x-www-form-urlencoded' \
     --header 'Authorization: Bearer
eyJ4NXQjUzI1NiI6ImZLS1Q5R2wyYXMwRThXOHg4SHNpaVdrdUJMS3R0TTlJRzJJY3U1M2RTSmsiLCJ4NXQiOiJqRzQ2Qzd0Nzc2YUY5eDRVY3dEODdM
ZnhYN2siLCJraWQiOiJTSUdOSU5HX0tFWSIsImFsZyI6IlJTMjU2In0.eyJzdWIiOiJmNzhmY2MxZmNhYmU0ODc1YmU1MmE0MDhhNTI5YzZjNyIsImd0
cCI6ImNjIiwidXNlci50ZW5hbnQubmFtZSI6ImlkY3MtODA2ZDNjY2E5NGRiNDljOWI4ZDFhZWFjYzdkODg5ZTAiLCJzdWJfbWFwcGluZ2F0dHIiOiJ1
c2VyTmFtZSIsInByaW1UZW5hbnQiOnRydWUsImlzcyI6IOlwvXC9pZGVudS5vcmFjbGVjZC5jb21cLyIsIna190eXBlIjoiQVQiLCJjbGllbnRfaWQiO
iJmNzhmY2MxZmNhYmU0ODc1YmU1MmE0MDhhNTI5YzZjNyIsImNhX2d1aWQiOiJjYWNjdC1jZTNlNjYxNzI1YzM0NGNiYTY0ZWFlZmM4ZjQ2NmQzNCIsI
mF6hdw9wjkd9eIHjiZko90897NJ90znZizOlwvXC9oaWx4YzJqZ3FibWxkenBlc3I3YzR0d2lraS5hcGlnYXRld2F5LnVzLWFzaGJ1cm4tMS5vY2kuY3
VzdG9tZXItb2NpLmNvbVwvIiwic3ViX3R5cGUiOiJjbGllbnQiLCJzY29wZSI6ImJpY2Nsb2dcL3JlcG9ydFN0YXRzUkVTVCIsImNsaWVudF90ZW5hbn
RuYW1lIjoiaWRjcy04MDZkM2NjYTk0ZGI0OWM5YjhkMWFlYWNjN2Q4ODllMCIsImV4cCI6MTYxMzE2NTM5MSwiaWF0IjoxNjEzMTYxNzkxLCJ0ZW5hbn
RfaXNzIjoiaHR0cHM6XC9cL2lkY3MtODA2ZDNjY2E5NGRiNDljOWI4ZDFhZWFjYzdkODg5ZTAuaWRlbnRpdHkub3JhY2xlY2xvdWQuY29tIiwiY2xpZW
50X2d1aWQiOiJiNzQyODlkY2I3Nzg0OWU4YWIwNjY5MjZlY2I2Y2QzMCIsImNsaWVudF9uYW1lIjoidWphbmtlLWFzc2V0LXRlc3QiLCJ0ZW5hbnQiOi
JpZGNzLTgwNmQzY2NhOTRkYjQ5YzliOGQxYWVhY2M3ZDg4OWUwIiwicmVzb3VyY2VfYXBwX2lkIjoiYjc0Mjg5ZGNiNzc4NDllOGFiMDY2OTI2ZWNiNm
NkMzAiLCJqdGkiOiIxMWViNmQ3MTBmNDQwNTYyYWMyZmNmY2M4M2VmNGI2NCJ9.Ejh-
X0YA5EjRJOwSgm86CBiJeNKhdOLMhoEWNvxyxiWY1mJVHJ1a3RP-gVQsd65WcjP2UoTelYV8e4af5-
SuJOW4h0IacnrADNAcAax1JDcUxpc8rsdptc8_6yitFlsx_POOe_mAuEIdp4hsmmCyxQBvPmP59F3aUaAZabOzckFDPjambQIbx_XGXMS5ZUFtUEQmII
G_koUgpv8m5XgDtjzVxU4mtbiYJ1XqPCXtT4EVQtJKyvgXI4MbAQURFOV3Cg1gLThZ0GYFIXXCSpRcxglasEc-
FJYQGkbMh9w2xODOgCyp2lCimaWcAPwAN0ESjofnfGnQPMkkHVEJa-DqPg'
```

```
   [{"oldestRunDate":"Oldest Extraction Date/Time","recentRunDate":"Latest Extraction
Date/Time","totalNumOfRowsInLogs":"Total Number or Extracted Rows Found","scheduleIDArr":[{"scheduleID":"List of
Batch Schedules Found"}],"requestIDArr":[{"requestID":"List of Jobs Found"}],"fileListArr":[{"fileName":"Processed
File List"}],"voListArr":[{"voName":"Processed VO List"}]},{"oldestRunDate":"2020-07-
13T10:00:06.323Z","recentRunDate":"2020-08-
06T18:05:04.965Z","totalNumOfRowsInLogs":"31363321","fileListArr":[{"fileName":"EXTRACT_STATUS_DATA_SCHEDULE_1663072
_REQUEST_1663073.JSON"},{"fileName":"EXTRACT_STATUS_PRIMARY_KEYS_SCHEDULE_1663072_REQUEST_1663073.JSON"},{"fileName"
:"EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663082.JSON"},{"fileName":"EXTRACT_STATUS_PRIMARY_KEYS_SCHEDULE_1663
072_REQUEST_1663082.JSON"},{"fileName":"EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663116.JSON"},{"fileName":"EXT
RACT_STATUS_PRIMARY_KEYS_SCHEDULE_1663072_REQUEST_1663116.JSON"},{"fileName":"EXTRACT_STATUS_DATA_SCHEDULE_1663072_R
EQUEST_1663152.JSON"},{"fileName":"EXTRACT_STATUS_PRIMARY_KEYS_SCHEDULE_1663072_REQUEST_1663152.JSON"},{"fileName":"
EXTRACT_STATUS_DATA_SCHEDULE_1663072_REQUEST_1663189.JSON"},{"fileName":"EXTRACT_STATUS_PRIMARY_KEYS_SCHEDULE_166307
2_REQUEST_1663189.JSON"}, ...
```

The tests as shown above ensure that we have entered a valid configuration. That means these protected API's can also be invoked from tools like Oracle Integration Cloud or Visual Builder once a user has signed into the OCI tenancy and given the required access to these API's via IDCS.

Conclusion

This article was meant to explain how an unprotected REST API as described in the previous blog can now be secured using standard OCI identity management capabilities.