Oracle® Communications

Network Analytics Automated Testing Suite Guide

Release 24.1.0
F92252-01
April 2024

Oracle Communications Network Analytics Automated Testing Suite Guide, Release 24.1.0

F92252-01

Contents

3 Installing ATS for Different Network Analytics Suite Products

My Oracle Support

My Oracle Support (https://support.oracle.com) is your initial point of contact for all product support and training needs. A representative at Customer Access Support can assist you with My Oracle Support registration.

Call the Customer Access Support main number at 1-800-223-1711 (toll-free in the US), or call the Oracle Support hotline for your local country from the list at http://www.oracle.com/us/support/contact/index.html. When calling, make the selections in the sequence shown below on the Support telephone menu:

- For Technical issues such as creating a new Service Request (SR), select **1**.
- For Non-technical issues such as registration or assistance with My Oracle Support, select **2**.
- For Hardware, Networking and Solaris Operating System Support, select **3**.

You are connected to a live agent who can assist you with My Oracle Support registration and opening a support ticket.

My Oracle Support is available 24 hours a day, 7 days a week, 365 days a year.

Acronyms

The following acronyms are used in the ATS User Guide.

Table Acronyms

Abbreviation	Definition
ATS	Automated Testing Suite
BDD	Behavior Driven Development. It is an agile software development technique that encourages collaboration between developers, QA, and non-technical or business participants in a software project.
CI	Continuous Implementation
CNE	Oracle Communications Cloud Native Core, Cloud Native Environment
OCNADD	Oracle Communications Network Analytics Data Director
OCNWDAF	Oracle Communications Network Data Analytics Function
OL	Oracle Linux is a Linux distribution packaged and freely distributed by Oracle, available partially under the GNU (General Public License) since late 2006. It is compiled from Red Hat Enterprise Linux source code, replacing Red Hat branding with Oracle's.

What's New in This Guide

This section introduces the documentation updates for Release 24.1.x in *Oracle Communications Network Analytics Automated Testing Suite Guide*.

Release 24.1.0 - F92252-01, April 2024

OCNADD Release 24.1.0

- Updated the following sections:
  - Updated the ATS Framework Features section to add the new ATS framework "TLS Support" feature supported by OCNADD.
  - Added a new section for Enable TLS Support on OCNADD ATS.
  - Added a new procedure for Enabling TLS.
  - Total number of resources, pod resource requirements updated in the Resource Requirements section.
  - OCNADD release versions updated in the Downloading the ATS Package section.
  - Updated the procedure for Pushing the Images to Customer Docker Registry.
  - The pipeline screen is updated in the OCNADD-NewFeatures Pipeline section.

OCNWDAF Release 24.1.0

Updated the following sections:

- OCNWDAF's directory package and folder structure is updated in the Downloading the ATS Package section.
- Image tags, release number and installers folder structure are updated in the Pushing the Images to Customer Docker Registry section.
- Image version updated in the Updating values.yaml File section.

1 Introduction

This document provides information about Automated Testing Suite (ATS) deployment model for the Network Analytics Suite products.

ATS allows you to run software test cases using an automated testing tool and then, compares the actual results with the expected or predicted results.

1.1 Overview

Using ATS, you can deploy and test 5G NFs.

This document provides information to implement ATS for the following 5G products:

- Oracle Communications Networks Data Analytics Function
- Oracle Communications Network Analytics Data Director

ATS for 5G Network Functions

For 5G NFs, ATS is developed using Oracle Linux 8-slim as the base image. Jenkins is a part of the ATS image, and it provides a graphical user interface (GUI) to test either a single NF or multiple NFs independently in the same network environment.

ATS comprises NF docker images, ATS image, simulator images, and test cases of the specific NF. All these images and test cases constitute a fully automated suite to deploy and test NFs. You can combine it with any other Continuous Integration (CI) pipeline with minimal changes because 5G ATS uses Jenkins as GUI.

The ATS package contains the following elements:

- Test scripts and docker images of test container. The docker images have complete framework and libraries installed, which are common for all NFs working with the Behavior Driven Development (BDD) framework.
- Docker image of HTTP Server simulator.
- Helm chart to deploy the ATS (delivered as a tar file).
- Readme text file (.txt file).

ATS provides basic environment, framework, and a GUI (Jenkins) to run all the functional test cases.

ATS Framework Version with Supporting NF Version

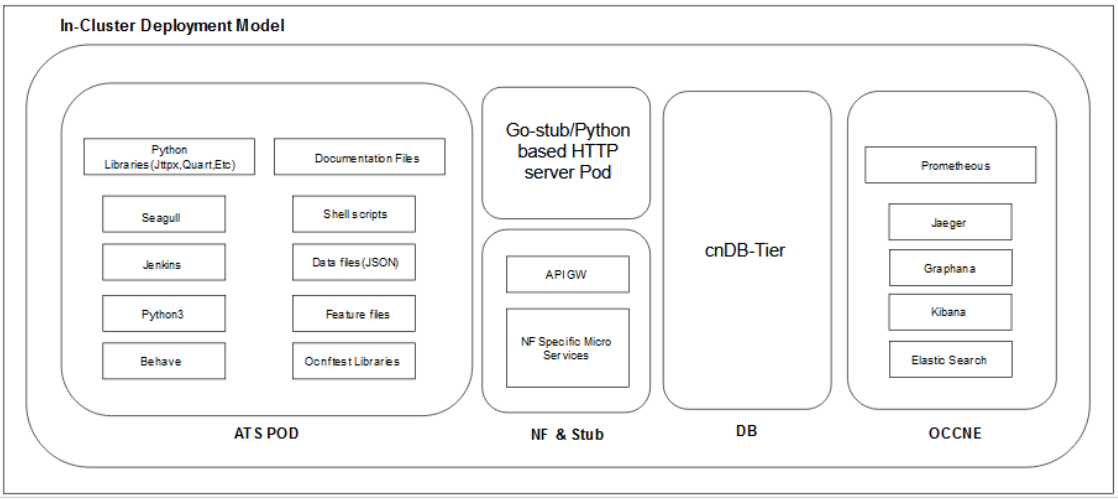
Table 1-1 ATS Framework Version with Supporting NF

ATS Framework Version	NWDAF	Data Director
24.1.0	24.1.0	NA
24.1.0	NA	24.1.0

1.2 Deployment Model

According to the In-Cluster deployment model, ATS can coexist in the same cluster where the NFs are deployed. This deployment model is useful for In-Cluster testing.

Figure 1-1 In-Cluster Deployment Model



1.3 References

For more information about NFs and their deployment processes, refer to the following documents:

- *Oracle Communications Network Data Analytics Function User Guide*
- *Oracle Communications Network Data Analytics Function Installation and Fault Recovery Guide*
- *Oracle Communications Network Analytics Data Director User Guide*
- *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*

2
ATS Framework Features

This chapter describes the ATS Framework features:

Table 2-1 ATS Framework Features Compliance Matrix

Features	NWDAF	OCNADD
Application Log Collection	Yes	No
ATS API	No	No
ATS Health Check	No	No
ATS Jenkins Job Queue	Yes	Yes
ATS Maintenance Scripts	Yes	Yes
ATS System Name and Version Display on Jenkins GUI	Yes	Yes
ATS Tagging Support	No	Yes
Custom Folder Implementation	Yes	No
Single Click Job Creation	Yes	Yes
Final Summary Report, Build Color, and Application Log	Yes	Partially compliant (Application Log is not supported.)
Lightweight Performance	Yes	No
Modifying Login Password	No	Yes
Parallel Test Execution	No	No
Parameterization	Yes	No
PCAP Log Collection	No	No
Persistent Volume	No	Optional
Test Result Analyzer	Yes	Yes
Test Case mapping and Count	Yes	Yes
TLS Support	No	Yes

Changes in Jenkins GUI

Listed below are some enhancements to the ATS Jenkins:

- Displays test cases under their stage and group names on the **BuildWithParameters** page of the ATS GUI.
- ATS Framework supports TLSv1.2 and TLSv1.3.
- ATS GUI Layout is enhanced.

2.1 ATS API

The Application Programming Interface (API) feature provides APIs to perform routine ATS tasks as follows:

- Start: To initiate one of the three test suites, such as Regression, New Features, or Performance.

- Monitor: To obtain the progress of a test suite's execution.
- Stop: To cancel an active test suite.
- Get Artifacts: To retrieve the JUNIT format XML test result files for a completed test suite.

For more information about configuring the tasks, see Use the RESTful Interfaces.

2.1.1 Creating API User and Granting Access

You require an account as an API user with access authorization and an API token that has to be generated for the user to perform routine ATS tasks using the Restful Interfaces API.

2.1.1.1 Creating an API User

Perform the following procedure to create an API user:

1. Log in to the ATS application using admin credentials.
2. In the left navigation pane of the ATS application, click **Manage ATS**.
3. Scroll down and click **Manage Users**.
4. In the left navigation pane, click **Create User**.
5. Enter the username as `<nf>apiuser`, for example, `policyapiuser`, `udrapiuser`, and a password.
6. The **Full name** field is optional. If left blank, it's automatically assigned a value by Jenkins.
7. Enter your email address as `<nf>apiuser@oracle.com`.
8. Click **Create User**. The API user is created.

2.1.1.2 Granting Access to the API User

Perform the following procedure to grant access to the API user:

1. In the left navigation pane, click **Manage ATS**.
2. Scroll down and click **Configure Global Security**.
3. Scroll down to **Authorization** and click **Add User**.
4. Enter the **username** created in the prompt that appears.
5. Check all the boxes in the Authorization matrix for apiuser that are also checked for `<nf>user`.
6. Click **Save**.
7. Go to the ATS main page and choose each of your NFs' pipeline.
8. In the left navigation pane, click **Configure**.
9. Scroll down to **Enable project-based security** and click **Add user.**
10. Enter the **user name** created in the prompt that appears.
11. Check all the boxes in the Authorization matrix for API user that are also checked for `<nf>user`.
12. Click **Save**.
    Now, API user can be used in API calls.

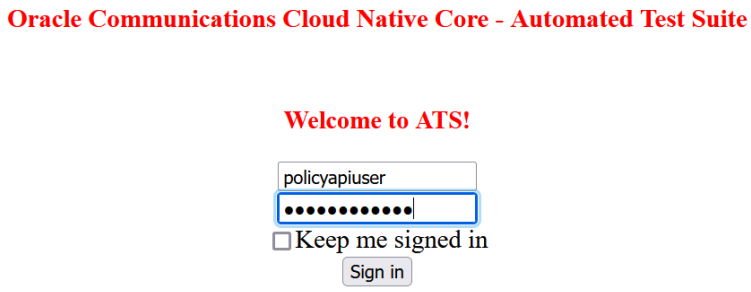
2.1.1.3 Generating an API Token for a User

Any API call requires the use of an API token for authentication. You can generate the API token, and it works until it is revoked or deleted.

Perform the following procedure to generate an API token for a user:

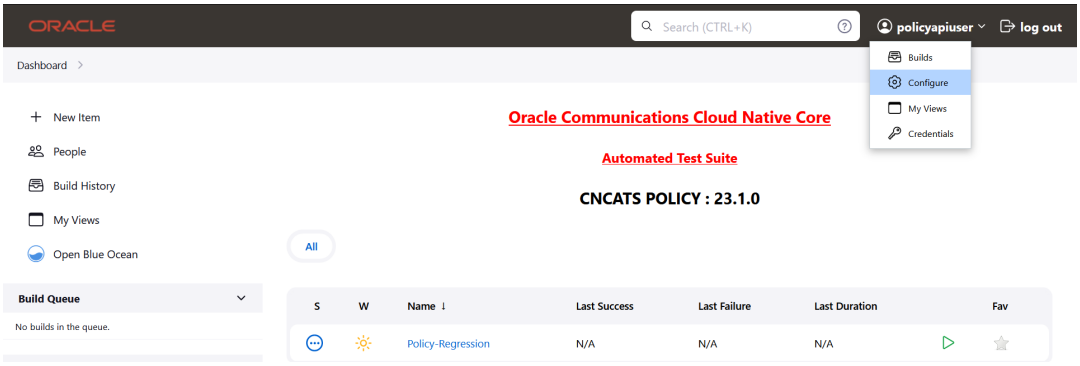
1. Log in to Jenkins as an NF API user to generate an API token.

Figure 2-1 ATS Login Page



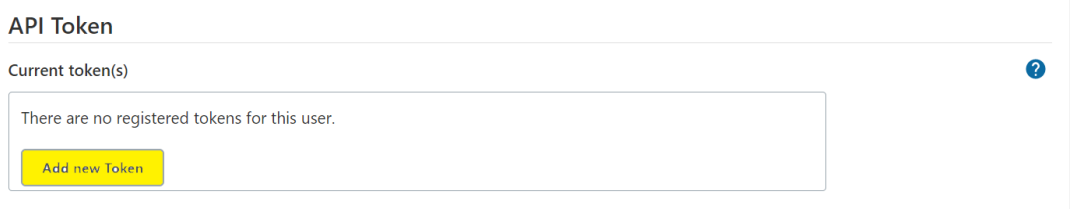
2. Click user name from the drop-down list at the top right of the Jenkins GUI, and then click **Configure**.

Figure 2-2 Configure to Add Token



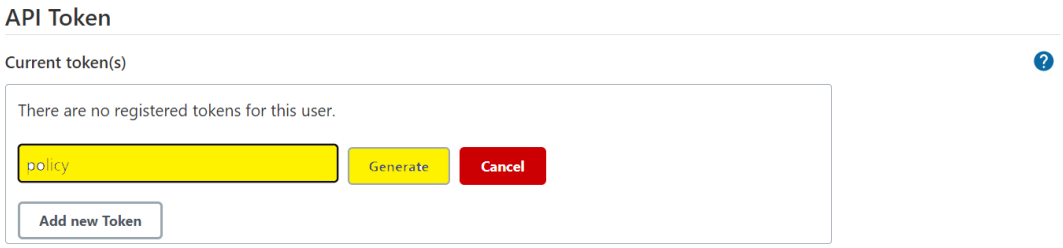
3. In the **API Token** section, click **Add New Token**.

Figure 2-3 Add New Token



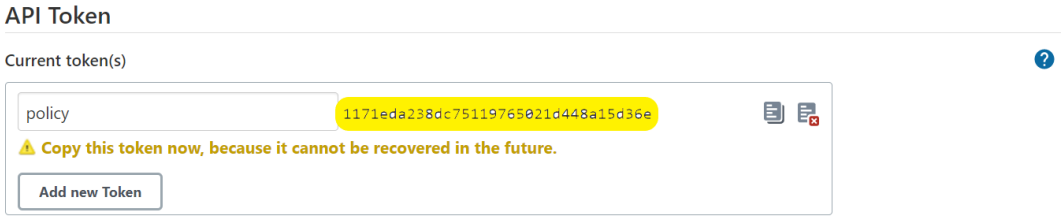
4. Enter a suitable name for the token, such as policy, and then click **Generate**.

Figure 2-4 Generate Token



5. Copy and save the generated token.
You cannot retrieve the token after closing the prompt.

Figure 2-5 Save Generated Token



6. Click **Save**.
An API token is generated and can be used for starting, monitoring, and stopping a job using the REST API.

2.1.2 Use the RESTful Interfaces

This section provides an overview of each RESTful interface.

2.1.2.1 Starting Jobs

To start a job, use the following RESTful interfaces:

- Default Jenkins API: The default Jenkins API to start a pipeline job
- Custom API: To start a job forcibly

Starting a Job using Default Jenkins API
If any other job is already running, the job started with this API goes to Jenkins' job queue.

- Run the following command to start a job (Default Jenkins method):

```
curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/
buildWithParameters –user
      <username>:<API_token> --verbose
```

The details of the parameters for the API are as follows:

Table 2-2 API Parameters

Parameters	Mandatory	Default Value	Description
userName	YES	NA	This parameter indicates the name of API user.
token	YES	NA	This parameter indicates the API token for API user.
Startjob_host_port	YES	NA	This parameter's format is `<host>:<port>` – `<host>` will be same as Jenkins host – `<port>` will be different(5001 or its nodeport)
pipelineName	YES	NA	This parameter indicates the name of the pipeline for which build is to be triggered.
pageAndQuery	YES	NA	This parameter can have two values: – buildWithParameters: for parametrized pipelines – build: for non-parametrized pipelines
jenkins_wait_time	NO	5	This parameter indicates the wait time for Jenkins in seconds. – If Jenkins is very slow in responding and the API response is not as expected, this wait time can be increased. – It is required when multiple running builds must be aborted before starting a new API build.
jenkins_host_port	YES	NA	This parameter's format is `<Jenkins host>:<Jenkins port>`.

For example,

```
curl --request POST http://10.123.154.163:30427/job/Policy-NewFeatures/
buildWithParameters
      --user policyapiuser:111ad02d7471cec9ca689696e9c7a55c62 --verbose
```

Starting a Job Forcibly using Custom API

If another job is already running and has not been started by an API user, the running job is aborted, along with all other jobs in the queue that have not been started by an API user, and a new job is started.

If the running job is started by the API user, the new job does not start, and the start job request fails, returning a message in response: `Build <job_id> of pipeline <pipeline_name> is already running, triggered by an API user`.

Builds are aborted gracefully by a forceful API, such as when a running scenario completes its execution and cleanup before the corresponding build is aborted.

The forceful API now returns an `aborted-builds` parameter in response, which contains job IDs for all the aborted builds. It also returns a parameter called `cancelled_builds_in_queue`, which contains queue IDs for all the builds aborted in queue.

If a job ID is assigned to a build in queue, it contains a list of two values: [queueid, jobid] rather than just the queue ID.

Run the following command to start a job forcibly:

```
curl -s --request POST <Startjob_host_port>/build -H "Content-Type:
application/json" –d
      '{"jenkins_host_port": "<Jenkins_host_port>", "pipelineName":
"<Pipeline_name>",
      "pageAndQuery": "<pageAndQuery>", "userName": "<username>", "token":
      "<API_token>"}' --verbose
```

For example,

```
curl --request POST 10.123.154.163:31423/build -H "Content-Type: application/
json" –d
      '{"jenkins_host_port": "10.123.154.163:30427", "pipelineName": "Policy-
NewFeatures",
      "pageAndQuery": "buildWithParameters", "userName": "policyapiuser",
"token":
      "111ad02d7471cec9ca689696e9c7a55c62"}' --verbose
```

> **Note**
>
> Startjob_host_port has the same IP as jenkins_host_port, but its port is different (the StartAPI port). ATS has two pairs of ports:
>
> - Jenkins port (8080) with its nodeport
> - StartAPI port (5001) with its nodeport

Customizing Job Parameters

Both of the Start APIs start the pipeline job with default parameter values. You can provide a different value for a parameter in an API call, such as `paramx=valuex`.

1. Append `paramx=value` to `buildWithParameters?`.
   Example 1,

   ```
   curl --request POST 10.75.217.40:31378/job/Policy-NewFeatures/
   buildWithParameters?paramx=valuex --user
         policyapiuser:110ed65222b9e63445689314998ff8c3bk -- verbose
   ```

   Example 2,

   ```
   curl --request POST 10.75.217.4:32476/build -H "Content-Type: application/
   json" -d '{"jenkins_host_port": "10.75.217.40:31378",
         "pipelineName": "Policy-NewFeatures", "pageAndQuery":
   "buildWithParameters?paramx=valuex", "userName":
         "policyapiuser", "token": "110ed65222b9e63445689314998ff8c3bk"}'
       –verbose
   ```

2. To add more than 1 parameter, such as `paramx=valuex` and `paramy=valuey`, append the other parameters to the API call using `&`.

Example 1,

```
curl --request POST 10.75.217.40:31378/job/Policy-NewFeatures/
buildWithParameters?paramx=valuex&paramy=valuey
--user policyapiuser:110ed65222b9e63445689314998ff8c3bk -- verbose
```

Example 2,

```
curl --request POST 10.75.217.4:32476/build -H "Content-Type: application/
json" -d '{"jenkins_host_port": "10.75.217.40:31378", "pipelineName":
"Policy-NewFeatures", "pageAndQuery": "buildWithParameters?
paramx=valuex&paramy=valuey", "userName": "policyapiuser", "token":
"110ed65222b9e63445689314998ff8c3bk"}' --verbose
```

3. Replace `buildWithParameters?` with `build` for non-parametrized pipeline jobs.
4. Start the pipeline by using the default Jenkins API or by changing the pageAndQuery parameter's value to build in the following way:

```
curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/build --user
<username>:<API_token> --verbose
```

Example 1,

```
curl --request POST 10.75.217.40:31378/job/Policy-NewFeatures/build --user
policyapiuser:110ed65222b9e63445689314998ff8c3bk --
      verbose
```

Example 2,

```
curl --request POST 10.75.217.4:32476/build -H "Content-Type: application/
json" -d '{"jenkins_host_port": "10.75.217.40:31378",
      "pipelineName": "Policy-NewFeatures", "pageAndQuery": "build",
"userName": "policyapiuser", "token":
      "110ed65222b9e63445689314998ff8c3bk"}' –verbose
```

2.1.2.2 Monitoring Jobs

This Default Jenkins API is used to monitor the progress of the job that was started.

For monitoring, the following APIs are used:

- A `qid` is obtained from the Location header in the response for `starting a job`. The first API uses this `qid` to get queue status about the corresponding job, including its `job_id`.
- The second API uses the `job_id` to obtain further information about the job status.

Monitoring a Job
To monitor jobs, run the following commands in a sequence:

1. `curl --request POST <Jenkins_host_port>/queue/item/<qid>/api/json --user <username>:<API_token> --verbose`

For example,

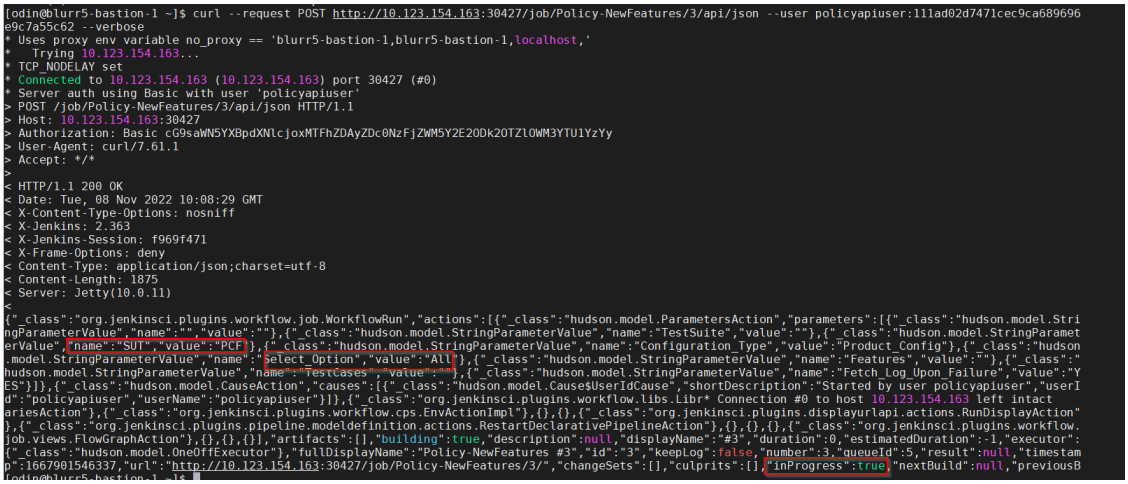
```
curl --request POST http://10.123.154.163:30427/queue/item/5/api/json--
user policyapiuser:111ad02d7471cec9ca689696e9c7a55c62
          --verbose
```

2. `curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/api/json --user <username>:<API_token> --verbose`
   For example,

```
curl --request POST http://10.123.154.163:30427/job/Policy-
NewFeatures/3/api/json--user
policyapiuser:111ad02d7471cec9ca689696e9c7a55c62
          --verbose
```

The following screenshot shows an example of monitoring the progress of the job:

Figure 2-6 Monitoring a Job



2.1.2.3 Stopping Jobs

Stop API is used to stop the currently running job using its job_id. It is also a default Jenkins API.

Stopping a Job

For ATS without Parallel Test Execution framework integrated:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/stop  --user
<username>:<API_token>
    --verbose
```

For example,

```
curl --request POST http://10.75.217.4:31881/job/UDR-Regression/21/stop --user
    udrapiuser:1139a72213e0a686972cbff4a2f9333a9f --verbose
```

> **Note**
>
> - If the rerun count is greater than zero, the job must be stopped twice.
> - This Stop API call does not abort the build gracefully.

For ATS with Parallel Test Execution framework integrated:

```
curl --request POST
      <Stopjob_host_port>/job/<Pipeline_name>/<job_id>/stop  --user
<username>:<API_token>
    --verbose
```

For example.

```
curl --request POST http://10.75.217.4:32476/job/UDR-Regression/21/stop --user
    udrapiuser:1139a72213e0a686972cbff4a2f9333a9f --verbose
```

The following table lists the parameter details for Stop API:

Table 2-3 Stop API Details

Parameter	Mandatory	Default Value	Description
userName	Yes	NA	Name of API user
token	Yes	NA	The API token for the API user
Stopjob_host_port	Yes	NA	Format is <host>:<port> • <host> is same as Jenkins host • <port> is different such as 5001 or its nodeport
pipelineName	Yes	NA	Name of the pipeline for which build is to be stopped

Table 2-3 (Cont.) Stop API Details

Parameter	Mandatory	Default Value	Description

immediate	No	False	To stop the build immediately, send a query parameter ("immediate=true") with API call.
			For example,
			
			`curl --request POST`
			
			`<Stopjob_host_port>/job/<Pipeline_name>/<job_id>/stop?immediate=true --user`
			
			`<username>:<API_token> --verbose`
			
			• `immediate` can also have values such as yes or 1. These values work similar to the true value.

2.1.2.4 Getting Test Suite Artifacts

Default Jenkins API is used to get the JUNIT-formatted XML test result files for a completed test suite.

For getting artifacts for any completed test suite, the following APIs are used:

- For getting an overall build summary
- For getting a JUNIT XML test result file for every feature file that ran

For getting an overall build summary:

```
curl --request POST <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/
testReport/api/xml?exclude=testResult/suite --user <username>:<API_token> --
verbose
```

For example,

```
curl --request POST http://10.123.154.163:30427/job/Policy-NewFeatures/4/
testReport/api/xml?exclude=testResult/suite--user
policyapiuser:111ad02d7471cec9ca689696e9c7a55c62 --verbose
```

For getting Feature-wise XML, Select_Option = All:

```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/artifact/test-results/
reports/*zip*/test-results.zip
      --user <username>:<API_token> --verbose --output  test-results.zip
```

For example,

```
curl --request POST http://
        10.75.217.4:31881/job/Policy-NewFeatures/21/artifact/test-results/
reports/*zip*/test-results.zip  --user
policyapiuser:11c3344996c4fda01ded2124bec4f9aa17
      --verbose –-output test-results.zip
```

For getting Feature-wise XML, Select_Option = Single/MultipleFeatures:

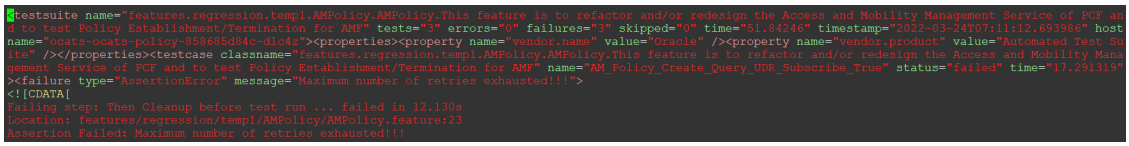
```
curl --request POST
      <Jenkins_host_port>/job/<Pipeline_name>/<job_id>/artifact/test-results/
reports/
      *.<Feature1_name>.xml,*.<Feature2_name>.xml/*zip*/test-results.zip  --
user <username>:<API_token>
      --verbose --output  test-results.zip
```

For example,

```
curl --request POST http://
      10.75.217.4:31881/job/Policy-NewFeatures/21/artifact/test-results/
reports/*.goldenfeature.xml,*.AMPolicy.xml/*zip*/test-results.zip
      –user policyapiuser:11c3344996c4fda01ded2124bec4f9aa17 --verbose  --
output test-results.zip
```

API calls for `Select_Option = All and Select_Option = Single/MultipleFeatures` return a zip file with JUNIT XMLs, one XML for each feature.

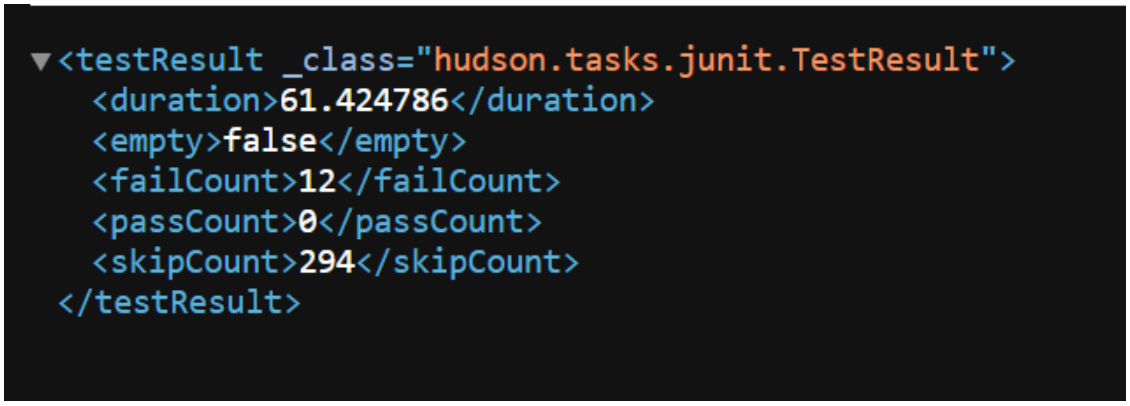
Figure 2-7 Sample XML Output for AMPolicy.feature



In the API call, specify other selected features in comma-separated form as `/*<Feature1_name>.xml,*<Feature2_name>.xml,*<Feature3_name>.xml,*<Feature4_name>.xml/` for Select_Option = Single/MultipleFeatures.

The API call for getting the overall build summary returns an XML with values for `duration`, `failCount`, `passCount`, and `skipCount` for the current build.

Figure 2-8 Sample Output



It is recommended to maintain a gap of at least a few seconds between two API calls. This gap depends on the time Jenkins takes to complete the API request.

2.2 ATS Health Check

ATS Health Check functionality is to check the health of the System Under Test (SUT)

Earlier, ATS used Helm test functionality to check the health of the System Under Test (SUT). With the implementation of the ATS Health Check pipeline, the SUT health check process has been automated. ATS health checks can be performed on webscale and non-webscale environments.

Convert a Value in Base64

The following command can be utilized to convert any value into base64 encoding:

```
echo-n "value"| base64
```

For example,

```
echo-n "126.98.76.43"| base64
```

Deploying ATS Health Check in a Webscale Environment

- Set the `Webscale` to 'true' and the following parameters by encoding them with base64 in the ATS values.yaml file:
- Set the following parameter to encrypted data:

```
webscalejumpserverip: encrypted-data
webscalejumpserverusername: encrypted-data
webscalejumpserverpassword: encrypted-data
webscaleprojectname: encrypted-data
webscalelabserverFQDN: encrypted-data
webscalelabserverport: encrypted-data
webscalelabserverusername: encrypted-data
webscalelabserverpassword: encrypted-data
```

Encrypted data is the value of parameters encrypted in base64. Fundamentally, Base64 is used to encode the parameters.

For example:

```
webscalejumpserverip=$(echo -n '10.75.217.42' | base64), Where Webscale Jump
server ip needs to be provided
webscalejumpserverusername=$(echo -n 'cloud-user' | base64), Where Webscale
Jump server Username needs to be provided
webscalejumpserverpassword=$(echo -n '****' | base64), Where Webscale Jump
server Password needs to be provided
webscaleprojectname=$(echo -n '****' | base64), Where Webscale Project Name
needs to be provided
webscalelabserverFQDN=$(echo -n '****' | base64), Where Webscale Lab Server
FQDN needs to be provided
webscalelabserverport=$(echo -n '****' | base64), Where Webscale Lab Server
Portneeds to be provided
webscalelabserverusername=$(echo -n '****' | base64), Where Webscale Lab
Server Username needs to be provided
webscalelabserverpassword=$(echo -n '****' | base64), Where Webscale Lab
Server Password needs to be provided
```

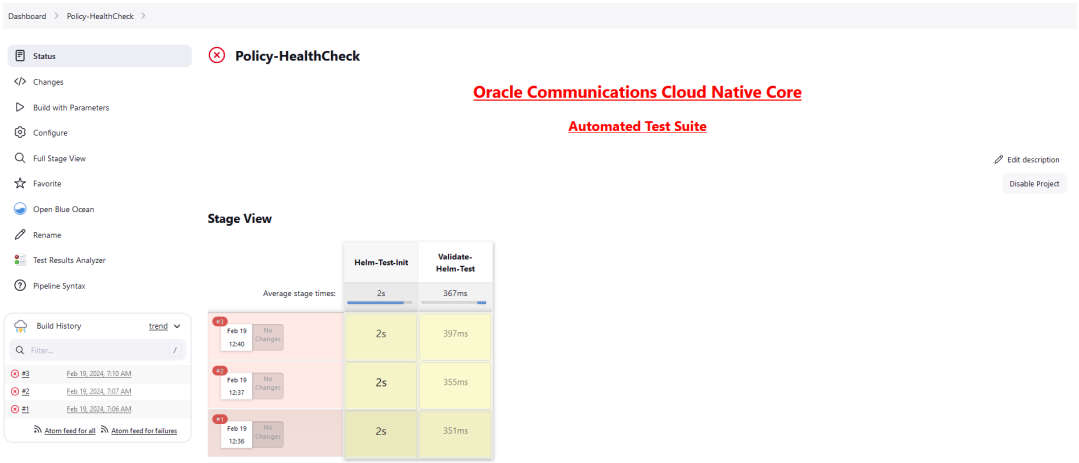
Running ATS Health Check Pipeline in an Webscale Environment

To run ATS Health Check pipeline:

1. Log in to ATS using respective <NF> login credentials.
2. Click <NF>HealthCheck pipeline and then click **Configure**.

> **Note**
>
> <NF> denotes the network function. For example, in Policy, it is called as Policy-HealthCheck pipeline.

Figure 2-9 Configure Healthcheck



3. Provide parameter a with Helm release name deployed. If there are multiple releases, use comma to provide all Helm release names.

```
//a = helm releases [Provide Release Name with Comma Separated if more
than 1 ]
```

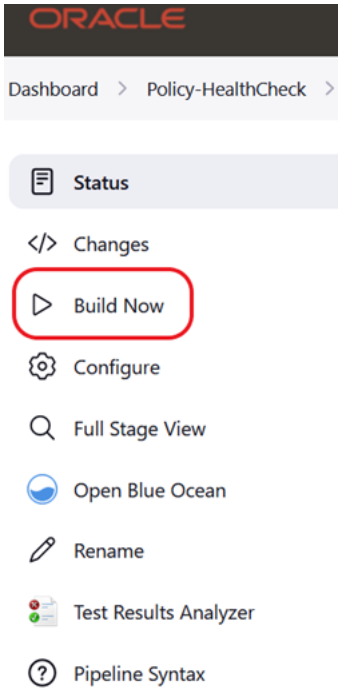
Provide parameter c with the appropriate Helm command, such as helm, helm3, or helm2.

```
//c = helm command name [helm or helm2 or helm3]
```

Figure 2-10 Save the Changes

4. Save the changes and click **Build Now**. ATS runs the health check on respective network function.

Figure 2-11 Build Now



Deploying ATS Health Check Pipeline in an OCI Environment

To use a ssh private key, create healthcheck-oci-secret and set the value of the key "passwordAuthenticationEnabled" to false.

Creating healthcheck-oci-secret

Create healthcheck-oci-secret to use ssh private keys instead of passwords using the following command:

```
kubectl create secret generic healthcheck-oci-secret --from-
file=bastion_key_file='<path of bastion ssh private key file>' --from-
file=operator_instance_key_file='<path of operator instance ssh private key
file>' -n <ATS namespace>
```

For example,

```
kubectl create secret generic healthcheck-oci-secret --from-
file=bastion_key_file='/tmp/bastion_private_key' --from-
file=operator_instance_key_file='/tmp/operator_instance_private_key' -n
seppsvc
```

> **Note**
>
> - Maintain the name of the secret as "healthcheck-oci-secret".
> - Ensure that the '--from-file' keys retain the same names: "bastion_key_file" and "operator_instance_key_file".
> - If the SSH private key is identical for both the bastion and operator instance, you can use the same path for both in the secret creation command.

Perform the following procedure to deploy ATS Health Check in a OCI environment:

Set the `Webscale` parameter set to 'false' and following parameters by encoding it with base64 in the ATS values.yaml file.

- To use password, provide base64 encoded values for key "password" for both bastion and operator instances, and set the value of key **passwordAuthenticationEnabled** to "true".
- Set the following parameter to encrypted data:

```
envtype: encrypted-data
ociHealthCheck:
  passwordAuthenticationEnabled: true or false
  bastion:
    ip: encrypted-data
    username: encrypted-data
    password: encrypted-data
  operatorInstance:
    ip: encrypted-data
    username: encrypted-data
    password: encrypted-data
```

> **Note**
>
> All fields are mandatory except for passwords. When the "passwordAuthenticationEnabled" field is set to true, only the "password" field needs to be updated; otherwise, it can remain with its default value.

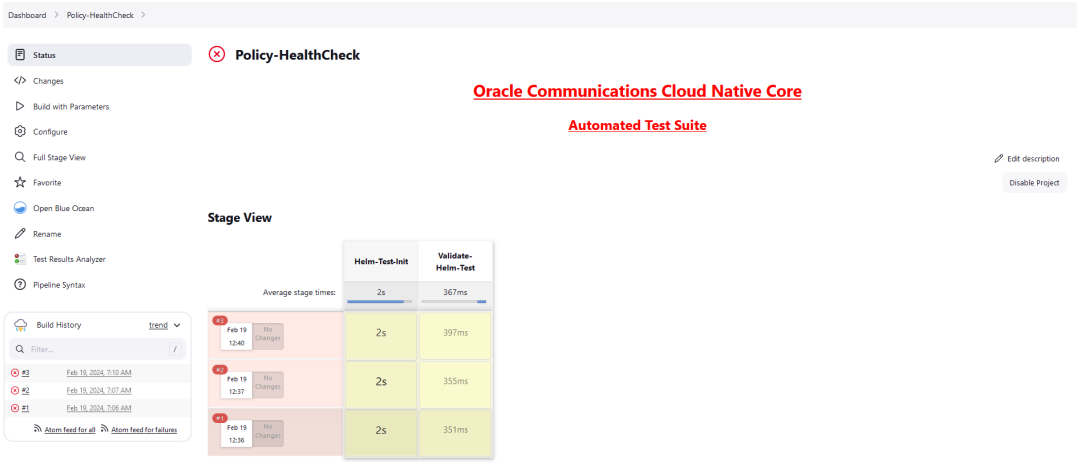
Running ATS Health Check Pipeline in an OCI Environment

To run ATS Health Check pipeline:

1. Log in to ATS using respective <NF> login credentials.
2. Click <NF>HealthCheck pipeline and then click **Configure**.

> **Note**
>
> <NF> denotes the network function. For example, in Policy, it is called as Policy-HealthCheck pipeline.

Figure 2-12 Configure Healthcheck



3. Provide parameter a with Helm release name deployed. If there are multiple releases, use comma to provide all Helm release names.

```
//a = helm releases [Provide Release Name with Comma Separated if more
than 1 ]
```

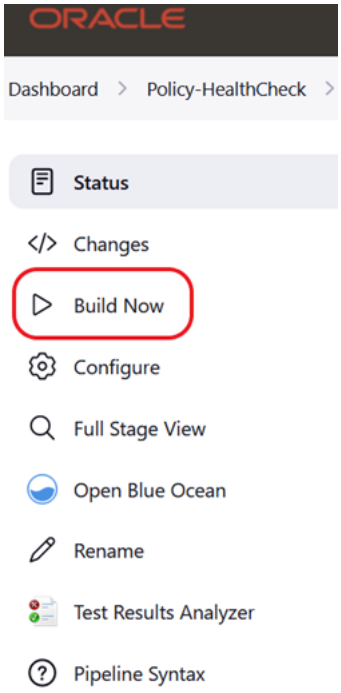
Provide parameter c with the appropriate Helm command, such as helm, helm3, or helm2.

```
//c = helm command name [helm or helm2 or helm3]
```

Figure 2-13 Save the Changes

4. Save the changes and click **Build Now**. ATS runs the health check on respective network function.

Figure 2-14 Build Now



Deploying ATS Health Check in a Non-Webscale or Non-OCI Environment

Perform the following procedure to deploy ATS Health Check in a non-webscale or non-OCI environment such as OCCNE:

Set the `Webscale` parameter set to 'false' and following parameters by encoding it with base64 in the ATS values.yaml file:

```
occnehostip: encrypted-data
occnehostusername: encrypted-data
occnehostpassword: encrypted-data
```

Example:

```
occnehostip=$(echo -n '10.75.217.42' | base64) , Where occne host ip needs to
be provided
occnehostusername=$(echo -n 'cloud-user' | base64), Where occne host username
needs to be provided
occnehostpassword=$(echo -n '****' | base64), Where password of host needs to
be provided
```

Running ATS Health Check Pipeline in a Non-Webscale or Non-OCI Environment

Perform the following procedure to run the ATS Health Check pipeline in a non-webscale or non-OCI environment such as OCCNE:

1. Log in to ATS using respective <NF> login credentials.
2. Click <NF>HealthCheck pipeline and then click **Configure**.
3. Provide parameter a with Helm release name deployed. If there are multiple releases, use comma to provide all Helm release names.

Provide parameter b with SUT deployed namespace name.

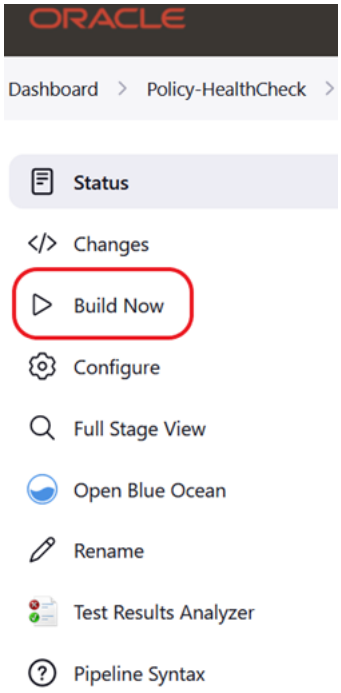
Provide parameter c with the appropriate Helm command, such as helm, helm3, or helm2.

```
//a = helm releases [Provide Release Name with Comma Separated if more
than 1 ]
//b = Namespace, If not applicable to WEBSCALE environment then remove the
argument
//c = helm command name [helm or helm2 or helm3]
```

Figure 2-15 Save the Changes

4. Save the changes and click **Build Now**. ATS runs the health check on respective network function.

Figure 2-16 Build Now



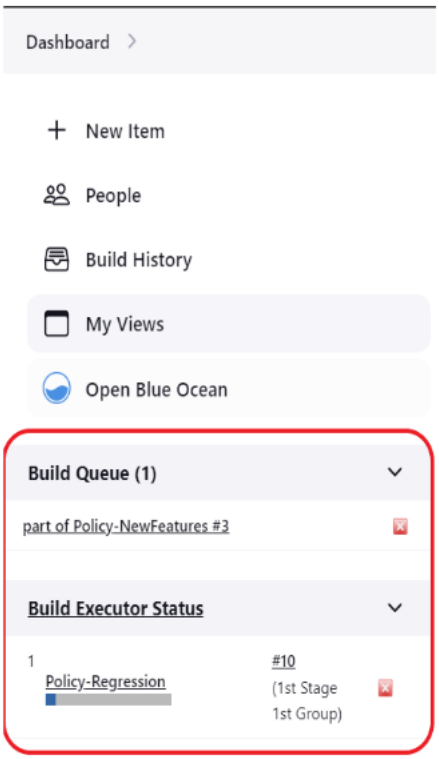
By clicking **Build Now**, you can run the health check on ATS and store the result in the console logs.

2.3 ATS Jenkins Job Queue

The ATS Jenkins Job Queue feature places the second job in a queue if the current job is already running from the same or different pipelines to prevent jobs from running in parallel to one another.

Job/build queue status can be viewed in the left navigation pane on the ATS home page. The following image shows the build queue status when a user has tried to run the NewFeatures pipeline when the Regression pipeline is already running.

Figure 2-17 Build Executor Status



2.4 Application Log Collection

Using Application Log Collection, you can debug a failed test case by collecting the application logs for NF System Under Test (SUT). Application logs are collected for the duration that the failed test case was run.

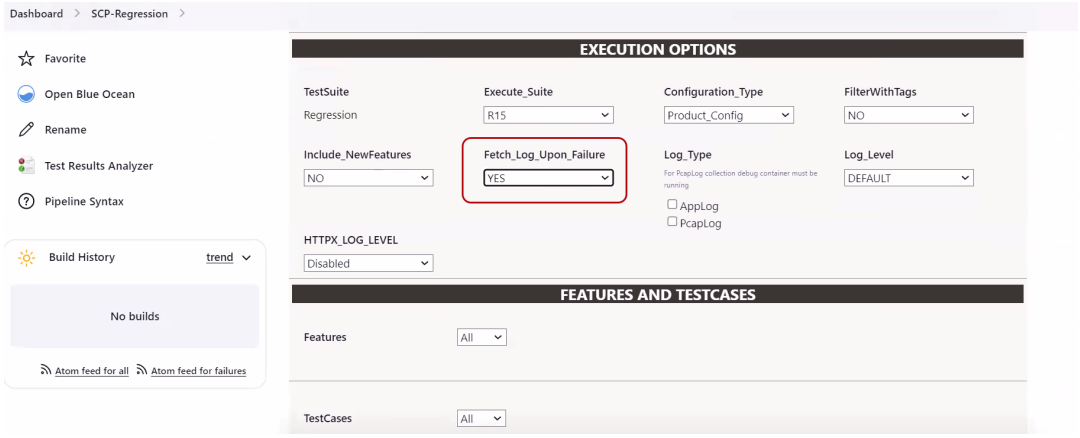
Application Log Collection can be implemented by using ElasticSearch or Kubernetes Logs. In both these implementations, logs are collected per scenario for the failed scenarios.

Application Log Collection Using ElasticSearch

To access the option to to collect logs using ElasticSearch:

1. Log in to ATS using respective <NF> login credentials.
2. On the NF home page, click any new feature or regression pipeline, from where you want to collect the logs.
3. In the left navigation pane, click **Build with Parameters**.
4. Select YES or NO from the drop-down menu of **Fetch_Log_Upon_Failure** to select whether the log collection is required for a particular run.

Figure 2-18 Fetch_Log_Upon_Failure



5. If option **Log_Type** is also available, select value **AppLog** for it.
6. Select the Log Level from the drop-down menu of Log_Level to set the log level for all the microservices. The possible values for Log_Level are as follows:
   - WARN: Designates potentially harmful situations.
   - INFO: Designates informational messages that highlight the progress of the application at coarse-grained level.
   - DEBUG: Designates fine-grained informational events that are most useful to debug an application.
   - ERROR: Designates error events that might still allow the application to continue running.
   - TRACE: The TRACE log level captures all the details about the behavior of the application. It is mostly diagnostic and is more granular and finer than DEBUG log level.

   > **Note**
   >
   > Log_Level values are NF dependent.

7. After the build execution is complete, go into the ATS pod, then navigate to following path to find the applogs:`.jenkins/jobs/<Pipeline Name>/builds/<build number>/` For example,`.jenkins/jobs/SCP-Regression/builds/5/`

   Applogs is present in zip form. Unzip it to get the log files.

The following tasks are carried out in the background to collect logs:

- ElasticSearch API is used to access and fetch logs.
- Logs are fetched from ElasticSearch for the failed scenarios
- Hooks (after scenario) within the cleanup file initiate an API call to Elasticsearch to fetch Application logs.
- Duration of the failed scenario is calculated based on the time stamp and passed as a parameter to fetch the logs from ElasticSearch.
- Filtered query is used to fetch the records based on Pod name, Service name, and timestamp (Failed Scenario Duration).

- For ElasticSearch, there is no rollover or rotation of logs over time.
- The maximum records that the ElasticSearch API can fetch per microservice in a failed scenario is limited to 10K.
- The following configuration parameters are used for collecting logs using Elastic Search:
  - ELK_WAIT_TIME: Wait time to connect to Elastic Search
  - ELK_HOST: Elastic Search HostName
  - ELK_PORT: Elastic Search Port

Application Log Collection Using Kubernetes Logs

To access the option to to collect logs using Kubernetes Logs:

1. On the NF home page, click any new feature or regression pipeline, from where you want to collect the logs.
2. In the left navigation pane, click **Build with Parameters**.
3. Select YES or NO from the drop-down menu of **Fetch_Log_Upon_Failure** to select whether the log collection is required for a particular run.
4. Select the Log Level from the drop-down menu of Log_Level to set the log level for all the microservices. The possible values for Log_Level are as follows:
   - WARN: Designates potentially harmful situations.
   - INFO: Designates informational messages that highlight the progress of the application at coarse-grained level.
   - DEBUG: Designates fine-grained informational events that are most useful to debug an application.
   - ERROR: Designates error events that might still allow the application to continue running.

> **Note**
>
> Log_Level values are NF dependent.

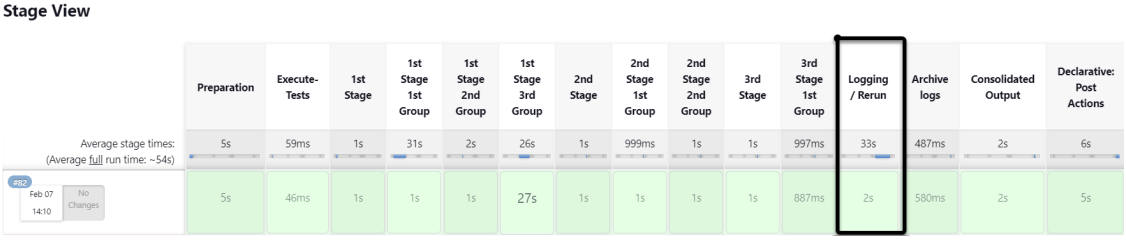
The following tasks are carried out in the background to collect logs:

- Kube API is used to access and fetch logs.
- For failed scenarios, logs are directly fetched from microservices.
- Hooks (after scenario) within the cleanup file initiate an API call to Elasticsearch to fetch Application logs.
- The duration of the failed scenario is calculated based on the time stamp and passed as a parameter to fetch the logs from microservices.
- Logs roll can occur while fetching the logs for a failed scenario. The maximum loss of logs is confined to a single scenario.

2.4.1 Application Log Collection and Parallel Test Execution Integration

A new stage,"Logging/Rerun", has been added at the end of the Execute-Tests stage to collect rerun logs, such as applog and PCAP logs, by running the failed test cases in a sequence.

Figure 2-19 Logging/Rerun new stage



If the `Fetch_Log_Upon_Failure` parameter is set to YES and if any test case fails in the initial run, then:

- The failed test case reruns and log collection start in the Logging/Rerun stage after the initial run is completed for all the test cases.
- The logs from the initial execution are collected, but they might be incorrect.
- Even if the `rerun` parameter is set to 0, the failed test case reruns in the Logging/Rerun stage and the log is collected.
- If the `Fetch_Log_Upon_Failure` parameter is set to NO and if any test case fails in the initial run, then the failed test case rerun starts in the same stage after the initial execution is over for all the test cases in its group.

2.5 ATS Maintenance Scripts

ATS maintenance scripts are used to perform the following operations:

- Taking a backup of the ATS custom folders and Jenkins pipeline.
- Viewing the configuration and restoring the Jenkins pipeline.
- Viewing the configuration and installing or uninstalling ATS and stubs.

ATS maintenance scripts are present in the ATS image at the following path: `/var/lib/jenkins/ocats_maint_scripts`

Run the following command to copy the scripts to a local system (bastion):

```
kubectl cp <NAMESPACE>/<POD_NAME>:/var/lib/jenkins/ocats_maint_scripts
<DESTINATION_PATH_ON_BASTION> pod
```

For example,

```
kubectl cp ocpcf/ocats-ocats-policy-694c589664-js267:/var/lib/Jenkins/
ocats_maint_scripts /home/meta-user/ocats_maint_scripts pod
```

2.5.1 ATS Scripts

ATS maintenance scripts are used to perform various task related to ATS and Jenkin pipeline.

The following are the types of scripts:

- **ats_backup.sh**: This script requires the user's input and takes a backup of the ATS custom folders, Jenkins jobs, and user's folders on the user's system. The backup can be of the Jenkins jobs and user's folder, the custom folders, or both. The custom folders

include cust_regression, cust_newfeatures, cust_performance, cust_data, and custom_config. For a Jenkins job or a user's folder, the script only takes a backup of the config.xml file. Also, the script requires the user to store a backup on the user's system (the default path is the location from where the script is being run) and to create a backup folder on the system and take the backup of the chosen folder from the corresponding ATS into the backup folder. The backup folder name can be of the following notation: `ats_<version>_backup_<date>_<time>`.

- **ats_uninstall.sh**: This script requires the user's input and uninstalls the corresponding ATS.
- **ats_install.sh**: This script requires the user's input and installs a new ATS. If `PVEnabled` is set to `true`, the script also reads the PVC name from values.yaml and creates values.yaml before installation. Also, if needed, the script performs the postinstallation steps, such as copying tests and Jenkins jobs' folders from the `ats_data` tar file to the ATS pod when PV is deployed, and then restarts the pod.
- **ats_restore.sh**: This script requires the user's inputs, restores the new release ATS pipeline, and views the configuration by referring to the last release ATS Jenkins jobs and the user's configuration. It depends on the user whether to use the backup folders from the user's system to restore the ATS configuration. If the user instructs the script to use the backup from the system, the script requires the path of the backup and uses the backup to restore. Otherwise, the script requires the last ATS Helm release name to refer to its Jenkins jobs and the user's configuration to restore.
  The script refers to the last release of ATS Jenkins pipelines and sets the `Discard old builds` property if this property is set in the last release of ATS for a pipeline but not in the current release. If this property is set in both releases, the script just updates the values according to the last release. Also, the script restores the `pipeline environment variables` values as per the last release of ATS. If any custom pipeline (created by the user) was present in the last release of ATS, the script restores that as well. It also restores the extra views created by NF users, for example, policy users, SCP users, and NRF users. Moreover, the script displays messages about the pending configuration that the user needs to perform manually. For example, a new pipeline or a new environment variable (for a pipeline) is introduced in the new release.

  While deploying ATS without PV, Jenkins needs to be restarted for the restore process to complete. If the last release ATS contains the `Configuration_Type` parameter, the `Configuration_Type` script needs to be approved with the **In Process Script Approval** setting under **Manage ATS** in Jenkins for the restore process to complete.

2.5.2 Updating ats_install.sh

Currently, the `ats_install.sh` script copies the tests folder and Jenkins jobs folder into the ATS pod and then restarts the pod when deployed with PV.

How to Update ats_install.sh

Other NFs can also use the `ats_install.sh` scripts. However, additional post installation steps may have to be performed manually for a few NFs.

For the additional post installation commands, perform the following steps:

1. In the `ats_install.sh` script, there is a post install section between `####POST_INSTALL_START####` and `#### POST_INSTALL_END ####`.
   a. Add the required post install commands.

> **Note**
>
> These commands are NF-specific.

b. Use the following commands:
   - `$namespace` for the namespace value
   - `$pod_name` for the pod name
   - `$ats_data_path` for the `ats_data` folder path (it has tests folder and Jenkins jobs folder provided as tar file in ATS package)

c. In the if-else block related to whether PV is enabled or not, add the following:
   - Add a command specific to `PVEnabled=true` in the `if block`.
   - Add a command specific to `PVEnabled=false` in the `else block`.

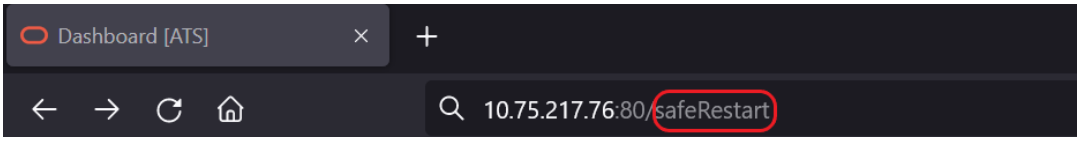
2. For additional inputs, enter the required code between `#### INPUT_START ####` and `#### INPUT_END ####`.

2.5.3 Restarting Jenkins without Restarting Pod

Perform the following procedure to restart Jenkins without restarting pods:

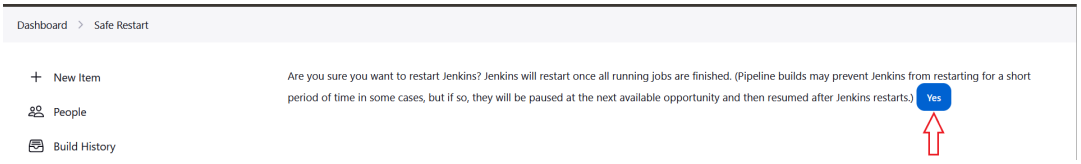
1. Log in as the Jenkins admin.
2. Go to the `<Jenkins_IP>:<port>/safeRestart`, for example, `10.87.73.32:32156/safeRestart`.

Figure 2-20 Safe Restart



3. Click **Yes**.

Figure 2-21 Restart Jenkins



2.5.4 Updating Stub Scripts

The following stubs can be updated:

- **stub_uninstall.sh**: This script requires the user's inputs and uninstalls all the stubs.
- **stub_install.sh**: This script requires the user's inputs and installs all the stubs.

> **Note**
>
> Currently, `stub_uninstall.sh` and `stub_install.sh` work.

Perform the following procedure to update the stub scripts for other NFs (NRF in this case):

1. Go to the `stub` folder.
2. From each script:
   a. Remove the CNC Policy-specific stubs inputs (dns, amf, and ldap), and add the input code blocks for NRF-specific stubs.
   b. For the stubs to uninstall, change the value of the `stubUninstallList` variable, and delete the variables for the CNC Policy-specific stubs below it.

      > **Note**
      >
      > `stubUninstallList` contains the Helm release names of the common stubs that are deployed generally.

   c. Declare the variables for the NRF-specific stubs below the `stubUninstallList` line.
   d. Remove the Helm uninstallation commands of the policy-specific stubs, and add the Helm uninstallation commands of the NRF-specific stubs.
   e. For the stubs to install, change the value of the `stubInstallList` variable, and delete the variables for the CNC Policy-specific stubs below it.

      > **Note**
      >
      > `stubInstallList` contains the Helm release names of the common stubs that are deployed generally.

   f. Declare the variables for the NRF-specific stubs below the `stubInstallList` line.
   g. Remove the Helm installation commands of the CNC Policy-specific stubs, and add the Helm installation commands of the NRF-specific stubs.

2.5.5 Running ATS and Stub Deployment Scripts

Perform the following procedure to run ATS and stub deployment scripts:

> **Note**
>
> If you want to take a backup of the custom folders or Jenkins jobs and user's configuration or both, run the `ats_backup.sh` script.

1. Run the `ats_install.sh` script to install the new release ATS (values.yaml of the ATS Helm chart must be updated before this step).
2. Run the `ats_restore.sh` script to restore the new ATS pipeline and view configuration.

> **Note**
>
> - You might perform the manual steps required for the restore script.
> - You must copy all the necessary changes to the new release ATS from the last release ATS. To get the changes in the last release, you must refer to the custom folders in the last release ATS backup on the system with an existing backup using `ats_backup.sh` before this step.
> - You can remove the last release ATS pod using the `ats_uninstall.sh` script while continuing to retain the last release PVC. You can use the last release PVC to port backward. Delete the last release PVC when you do not require the backward porting.

3. Run the `stub_install.sh` script to install all the new release stubs values.yaml of the stub Helm charts must be updated before this step.
4. Run the `stub_uninstall.sh` script to uninstall all the last release stubs.

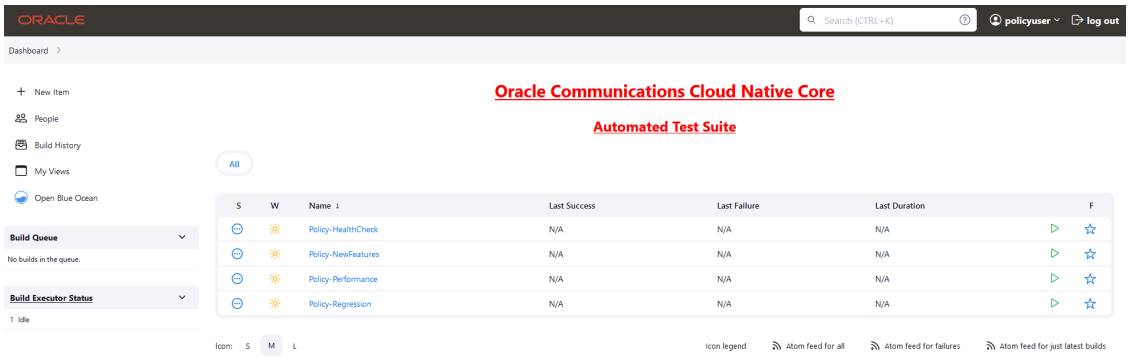
2.6 ATS System Name and Version Display on the ATS GUI

This feature displays the ATS system name and version on the ATS GUI.

You can log in to the ATS application using the login credentials to view the following:

- ATS system name: Abbreviated product name followed by NF name.
- ATS Version: Release version of ATS.

Figure 2-22 ATS System Name and Version



2.7 ATS Tagging Support

The ATS Tagging Support feature assists in running the feature files after filtering features and scenarios based on tags. Instead of manually navigating through several feature files, the user can save time by using this feature.

The GUI offers the following four options for selecting tag types:

- Feature_Include_Tags: The features that contain either of the tags available in the **Feature_Include_Tags** field are considered for tagging.
  - For example, "cne-common", "config-server". All the features that have either "cne-common" or "config-server" tags are taken into consideration.

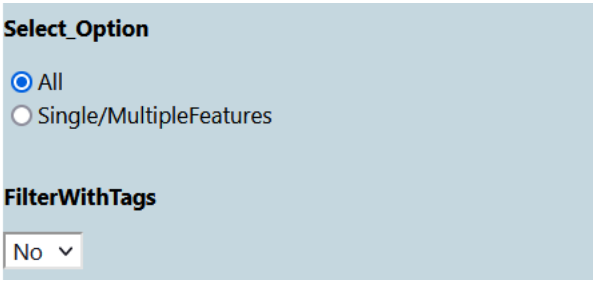
- Feature_Exclude_Tags: The features that contain neither of the tags available in the **Feature_Exclude_Tags** field are considered for tagging.
  - For example, "cne-common","config-server". All the features that have neither "cne-common" nor "config-server" as tags are taken into consideration.
- Scenario_Include_Tags: The scenarios that contain either of the tags available in the **Scenario_Include_Tags** field are considered.
  - For example, "sanity", "cleanup". The scenarios that have either "sanity" or "cleanup" tags are taken into consideration.
- Scenario_Exclude_Tags: The features that contain neither of the tags available in the **Scenario_Exclude_Tags** field are considered.
  - For example, "sanity", "cleanup". The scenarios that have neither "sanity" nor "cleanup" as tags are taken into consideration.

Filter with Tags

The procedure to filter feature files and scenarios based on tags are as follows:

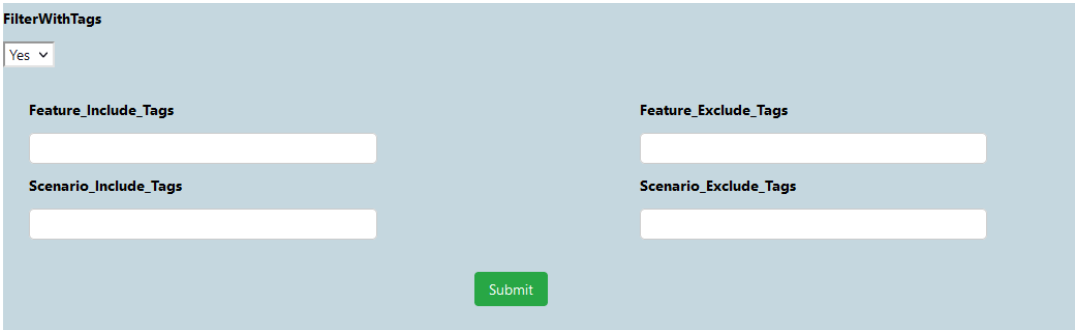
1. On the NF home page, click any new feature or regression pipeline, where you want to use this feature.
2. In the left navigation pane, click **Build with Parameters**. The following image appears.

Figure 2-23 Filter with Tags



3. Select **Yes** under **FilterWithTags**. The result shows four input fields.

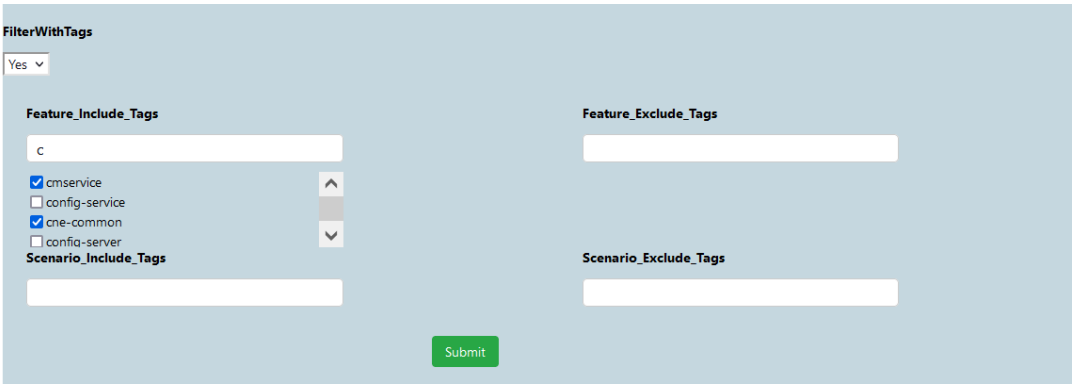
Figure 2-24 Types of Tags



The default value of **FilterWithTags** field is "No".

4. The input fields serve as a search or filter, displaying all tags that match the prefix entered. You can select one or multiple tags.

Figure 2-25 Tags Matching with Entered Prefix



5. Select the required tags from the different tags list and click **Submit**.

The specified feature-level tags are used to filter out features that contain any one of the include tags and none of the exclude tags. Here, any or both the fields may be left empty. All features are automatically taken into consideration when both fields are empty.

The scenario level tags are used to filter out the scenarios from the features filtered above. Only scenarios with any of the include tags and none of the exclude tags are considered. Any or both fields can be empty. When both fields are empty, all the scenarios from the above filtered feature files are considered.

> **Note**
>
> - If you select the Select_Option as 'All', all the displayed features and scenarios will run.
> - If you select the Select_Option as 'Single/MultipleFeatures, it enables you to select some features, and only those features and respective scenarios are going to run.

2.7.1 Combination of Tags and their Results

The combination of tags and expected results are as follows.

Table 2-4 Result of Filtered Tags

Feature_Include	Feature_Exclude	Scenario_Include	Scenario_Exclude	Results
-	-	-	-	All the features and scenarios are taken into consideration.
"abc","def"	-	-	-	Features with either "abc" or "def" tags and all scenarios from the filtered features are taken into consideration.

Table 2-4 (Cont.) Result of Filtered Tags

Feature_Include	Feature_Exclude	Scenario_Include	Scenario_Exclude	Results
-	"abc","def"	-	-	All the features with neither "abc" nor "def" tags and all scenarios from the filtered features are taken into consideration.
-	-	"sanity","cne"	-	Scenarios with either "sanity" or "cne" tags and features having these scenarios are taken into consideration.
-	-	-	"sanity","cne"	Scenarios with neither "sanity" nor "cne" tags and features having these filtered scenarios are taken into consideration.
"abc","def"	"ghi"	-	-	Features with either "abc" or "def" tags but without the "ghi" tag and all scenarios from filtered features are taken into consideration.
"abc","def"	-	"sanity","cne"	-	Scenarios only with either "sanity" or "cne" tags and only features that contain these scenarios and have either "abc" or "def" as feature tags are taken into consideration.
"abc","def"	-	-	"sanity","cne"	Scenarios with neither "sanity" nor "cne" tags and only features that contain the filtered scenarios and have either "abc" or "def" feature tags are taken into consideration.
-	"ghi"	"sanity","cne"	-	Features without the "ghi" tag and scenarios with either "sanity" or "cne" tags from the filtered features are taken into consideration.
-	"ghi"	-	"sanity","cne"	Features without the "ghi" tag and scenarios without the "sanity" and "cne" tags from filtered features are taken into consideration.

Table 2-4 (Cont.) Result of Filtered Tags

Feature_Include	Feature_Exclude	Scenario_Include	Scenario_Exclude	Results
-	-	"sanity","cne"	"cleanup"	Scenarios with either the "sanity" or "cne" tags and without the "cleanup" tag and features with filtered scenarios are taken into consideration.
"abc","def"	"ghi"	"sanity","cne"	-	Scenarios with either the "sanity" or "cne" tags and features that have these scenarios and have either the "abc" or "def" tags but not the "ghi" tag are taken into consideration.
"abc","def"	-	"sanity","cne"	"cleanup"	Scenarios with either the "sanity" or "cne" tags and without the "cleanup" tag, and features having the filtered scenarios and having the feature tags either "abc" or "def" are taken into consideration.
"abc","def"	"ghi"	-	"cleanup"	Scenarios without the tag "cleanup", and features with filtered scenarios and having either "abc" or "def" as feature tags but not the "ghi" tag are taken into consideration.
-	"ghi"	"sanity","cne"	"cleanup"	Scenarios with either the "sanity" or "cne" tags and without the "cleanup" tag, and features with filtered scenarios and not the tag "ghi," are taken into consideration.
"abc","def"	"ghi"	"sanity","cne"	"cleanup"	Scenarios with either "sanity" or "cne" tags and without the "cleanup" tag, and features with filtered scenarios and feature tags either "abc" or "def" but without the tag "ghi" are taken into consideration.

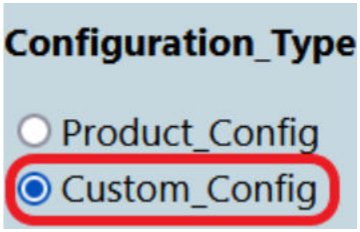
> **Note**
>
> The tags mentioned in the table are just examples; they may or may not be actually used.

2.8 Custom Folder Implementation

The Custom Folder Implementation feature allows the user to update, add, or delete test cases without affecting the original product test cases in the new features, regression, and performance folders. The implemented custom folders are cust_newfeatures, cust_regression, and cust_performance. The custom folders contain the newly created, customised test cases.

Initially, the product test case folders and custom test case folders will have the same set of test cases. The user can perform customization in the custom test case folders, and ATS always runs the test cases from the custom test case folders. If the option "Configuration_Type" is present on the GUI,the user needs to set its value to "Custom_Config" to populate test cases from the custom test case folders.

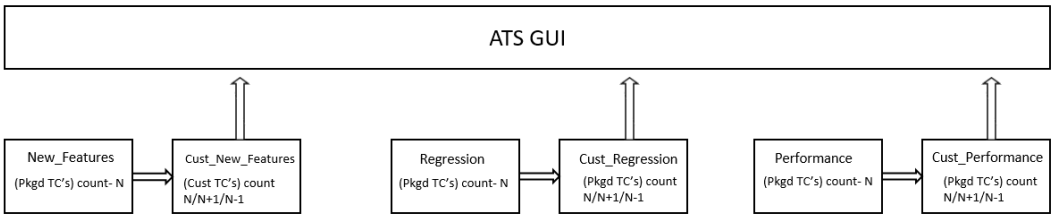
Figure 2-26 Custom Config Folder



Summary of Custom Folder Implementation

- Separate folders such as cust_newfeatures, cust_regression, and cust_performance are created to hold the custom cases.
- The prepackaged test cases are available in the newfeature and regression Folder.
- The user copies the required test cases to the cust_newfeatures and cust_regression folders, respectively.
- Jenkins always points to the cust_newfeatures and cust_regression folders to populate them in the menu.
  If someone initially launches ATS, they will not see any test cases in the menu if the cust folders are not populated. To avoid this, it is recommended to prepopulate both the folders, cust and original, and ask the user to modify only the cust folder if needed.

Figure 2-27 Summary of Custom Folder Implementation



2.9 Single Click Job Creation

With the help of Single Click Job Creation feature, ATS users can easily create a job to run TestSuite with a single click.

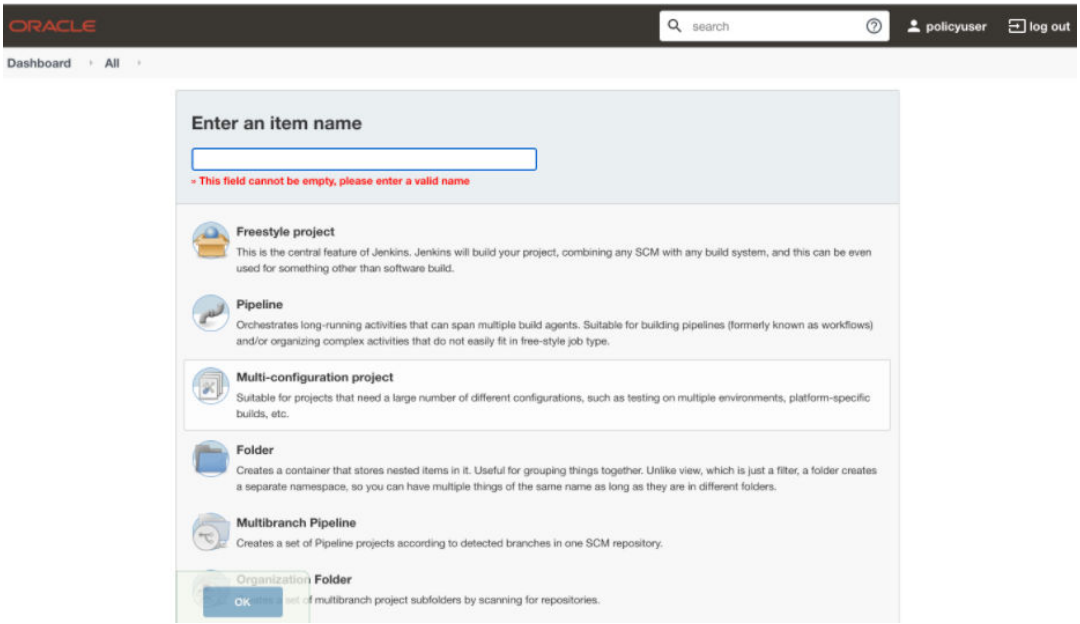
2.9.1 Configuring Single Click Job

Prerequisite: The network function specific user should have 'Create Job' access.

Perform the following procedure to configure the single-click feature:

1. Log in to ATS using network function specific log-in credentials..
2. Click **New Item** in the left navigation pane of the ATS application. The following page appears:

Figure 2-28 New Item Window



3. In the **Enter an item name** text box, enter the job name. Example: <NF-Specific-name>-NewFeatures.
4. In the **Copy from** text box, enter the actual job name for which you need single-click execution functionality. Example: <NF-Specific-name>-NewFeatures.

5. Click **OK**. You are automatically redirected to edit the newly created job's configuration.
6. Under the **General** group, deselect the **This Project is Parameterised** option.
7. Under the **Pipeline** group, make the corresponding changes to remove the 'Active Choice Parameters' dependency.
8. Provide the default values for the **TestSuite**, **SUT**, **Select_Option**, **Configuration_Type**, and other parameters, as required, on the **BuildWithParameters** page.
   Example: Pipeline without Active Choice Parameter Dependency

```
node ('built-in'){
    //a = SELECTED_NF    b = PCF_NAMESPACE        c = PROMSVC_NAME       d
= GOSTUB_NAMESPACE
    //e = SECURITY       f = PCF_NFINSTANCE_ID   g = POD_RESTART_TIME   h
= POLICY_TIME
    //i = NF_NOTIF_TIME  j = RERUN_COUNT         k =INITIALIZE_TEST_SUITE
l = STUB_RESPONSE_TO_BE_SET
    //m = POLICY_CONFIGURATION_ADDITION          n = POLICY_ADDITION
o = NEXT_MESSAGE
    //p = PROMSVCIP     q = PROMSVCPORT         r = TIME_INT_POD_DOWN    s
= POD_DOWN_RETRIES
    //t = TIME_INT_POD_UP   u = POD_UP_RETRIES  v = ELK_WAIT_TIME   w =
ELK_HOST
    //x = ELK_PORT  y = STUB_LOG_COLLECTION z = LOG_METHOD A =
enable_snapshot B = svc_cfg_to_be_read C = PCF_API_ROOT

    //Description of Variables:

    //SELECTED_NF : PCF
    //PCF_NAMESPACE : PCF Namespace
    //PROMSVC_NAME : Prometheus Server Service name
    //GOSTUB_NAMESPACE : Gostub namespace
    //SECURITY : secure or unsecure
    //PCF_NFINSTANCE_ID : nfInstanceId in PCF application-config config map
    //POD_RESTART_TIME : Greater or equal to 60
    //POLICY_TIME : Greater or equal to 120
    //NF_NOTIF_TIME : Greater or equal to 140
    //RERUN_COUNT : Rerun failing scenario count
    //TIME_INT_POD_DOWN : The interval after which we check the POD status
if its down
    //TIME_INT_POD_UP : The interval after which we check the POD status
if its UP
    //POD_DOWN_RETRIES : Number of retry attempt in which will check the
pod down status
    //POD_UP_RETRIES : Number of retry attempt in  which will check the
pod up status
    //ELK_WAIT_TIME : Wait time to connect to Elastic Search
    //ELK_HOST : Elastic Search HostName
    //ELK_PORT : Elastic Search Port
    //STUB_LOG_COLLECTION : To Enable/Disable Stub logs collection
    //LOG_METHOD : To select Log collection method either elasticsearch or
kubernetes
    //enable_snapshot: Enable or disable snapshots that are created at the
start and restored at the end of each test run
    //svc_cfg_to_be_read: Timer to wait for importing service
configurations
```

```
    //PCF_API_ROOT: PCF_API_ROOT information to set Ingress gateway
service name and port
    withEnv([
    'TestSuite=NewFeatures',
    'SUT=PCF',
    'Select_Option=All',
    'Configuration_Type=Custom_Config'
    ]){
    sh '''
        sh /var/lib/jenkins/ocpcf_tests/preTestConfig-NewFeatures-PCF.sh \
        -a PCF \
        -b ocpcf \
        -c occne-prometheus-server \
        -d ocpcf \
        -e unsecure \
        -f fe7d992b-0541-4c7d-ab84-c6d70b1b0123 \
        -g 60 \
        -h 120 \
        -i 140 \
        -j 2 \
                -k 0 \
                -l 1 \
                -m 1 \
                -n 15 \
                -o 1 \
                -p occne-prometheus-server.occne-infra\
                -q 80\
                -r 30\
                -s 5\
                -t 30\
                -u 5\
                -v 0\
                -w occne-elastic-elasticsearch-master.occne-infra\
                -x 9200\
                -y yes\
                -z kubernetes\
                -A no\
                -B 15\
                -C ocpcf-occnp-ingress-gateway:80\

    '''
    load "/var/lib/jenkins/ocpcf_tests/jenkinsData/Jenkinsfile-Policy-
NewFeatures"
   }
}
```

9. Click **Save**. The ATS application is ready to run TestSuite with 'SingleClick' using the newly created job.

2.10 Managing Final Summary Report, Build Color, and Application Log

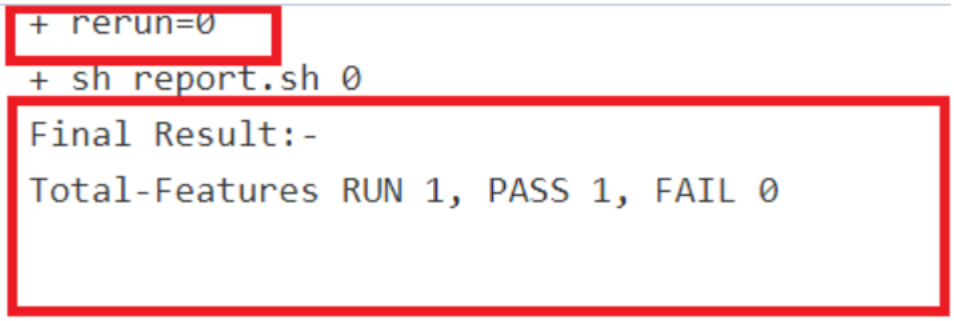
This feature displays an overall execution summary, such as the total run count, pass count, and fail count.

Supports Implementation of Total-Features

ATS supports implementation of **Total-Features** in the final summary report. Based on the rerun value set, the **Final Result** section in the final summary report displays the Total-Features output.

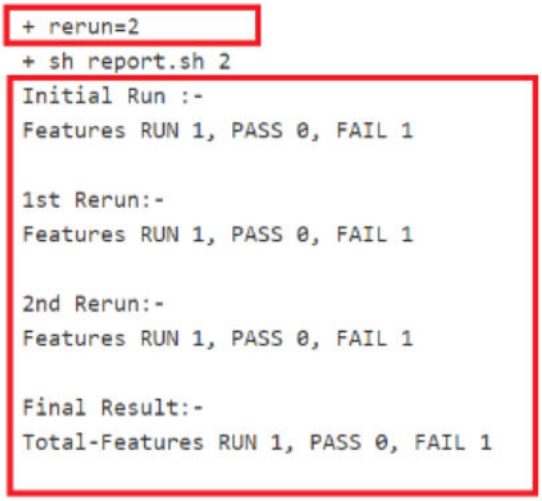
- If rerun is set to 0, the test result report shows the following result:

Figure 2-29 Total-Features = 1, and Rerun = 0



- If rerun is set to non-zero, the test result report shows the following result:

Figure 2-30 Total-Features = 1, and Rerun = 2



Changes After Parallel Test Execution Framework Feature Integration

After incorporating the Parallel Test Execution feature, the following results were obtained:

Final Summary Report Implementations

Figure 2-31 Group Wise Results

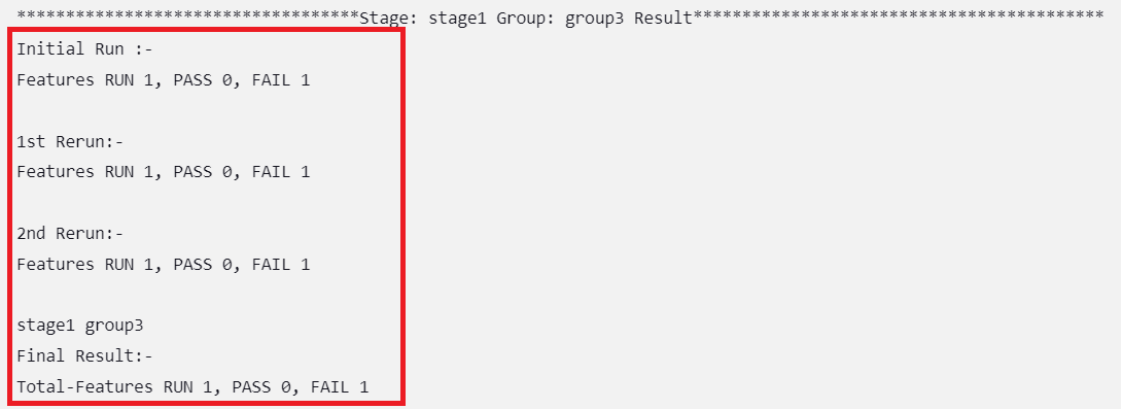


Figure 2-32 Overall Result When Selected Feature Tests Pass

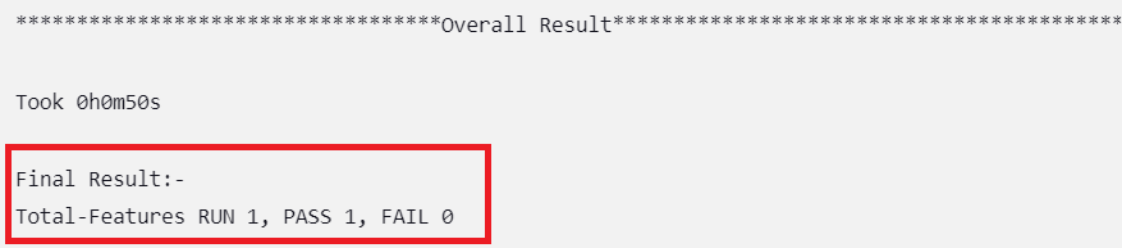
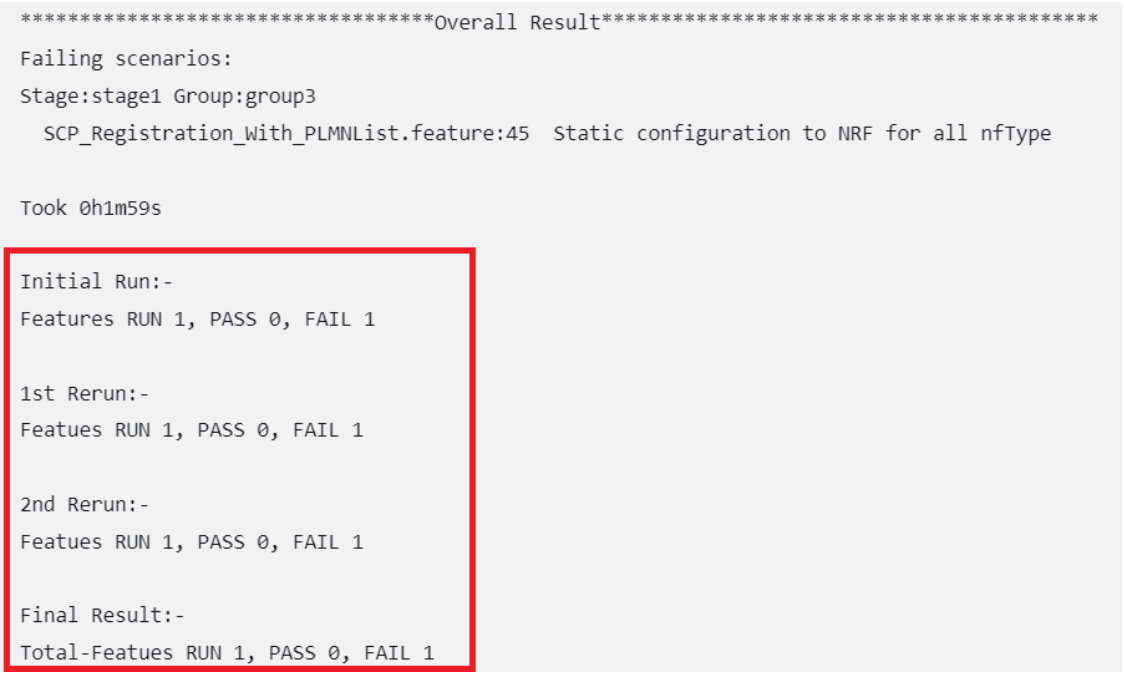


Figure 2-33 Overall Result When Any of the Selected Feature Tests Fail



Implementing Build Colors

ATS supports implementation of build color. The details are as follows:

Table 2-5 Build Color Details

Rerun Values	Rerun set to zero		Rerun set to non-zero		
Status of Run	**All Passed in Initial Run**	**Some Failed in Initial Run**	**All Passed in Initial Run**	**Some Passed in Initial Run, Rest Passed in Rerun**	**Some Passed in Initial Run, Some Failed Even After Rerun**
Build Status	SUCCESS	FAILURE	SUCCESS	SUCCESS	FAILURE
Pipeline Color	GREEN	Execution Stage where test cases failed shows YELLOW color, rest of the successful stages are GREEN.	GREEN	GREEN	Execution Stage where test cases failed shows YELLOW color, rest of the successful stages are GREEN
Status Color	BLUE	RED	BLUE	BLUE	RED

Changes After Integrating Parallel Test Execution Framework Feature
In sequential execution, the build color or overall pipeline status of any run was mainly dependent on the following parameters:

- the rerun count and the pass or fail status of test cases in the initial run
- the rerun count and the pass or fail status of test cases in the final run

For the parallel test case execution, the pipeline status also depends on another parameter, "Fetch_Log_Upon_Failure," which is given in the **build with parameters** page. If the parameter `Fetch_Log_Upon_Failure` is not there, its default value is considered "NO".

Table 2-6 Pipeline Status When Fetch_Log_Upon_Failure = NO

Rerun Values	Rerun set to zero		Rerun set to non-zero		
Passed/Failed	**All Passed in Initial Run**	**Some Failed in Initial Run**	**All Passed in Initial Run**	**Some Passed in Initial Run, Rest Passed in Rerun**	**Some Passed in Initial Run, Some Failed Even After Rerun**
Status	SUCCESS	FAILURE	SUCCESS	SUCCESS	FAILURE

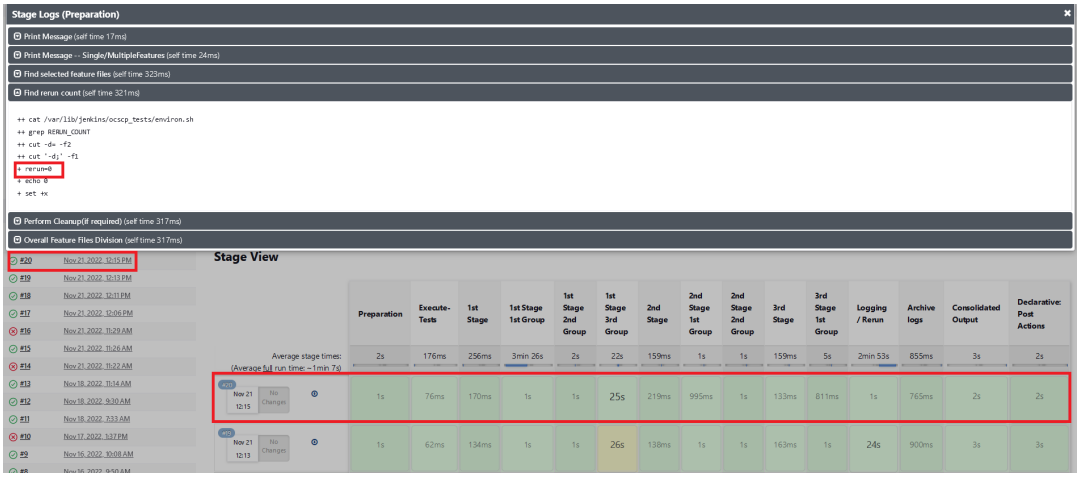
Table 2-7 Pipeline Status When Fetch_Log_Upon_Failure = YES

Rerun Values	Rerun set to zero			Rerun set to non-zero		
Passed/Failed	**All Passed in Initial Run**	**Some Failed in Initial Run and Failed in Rerun**	**Some Failed in Initial Run and Passed in Rerun**	**All Passed in Initial Run**	**Some Passed in Initial Run, Rest Passed in Rerun**	**Some Passed in Initial Run, Some Failed Even After Rerun**
Status	SUCCESS	FAILURE	SUCCESS	SUCCESS	SUCCESS	FAILURE

Some common combinations of these parameters, such as `rerun_count`, `Fetch_Log_Upon_Failure`, and `pass/fail status of test cases in initial and final run` and the corresponding build colors are as follows:

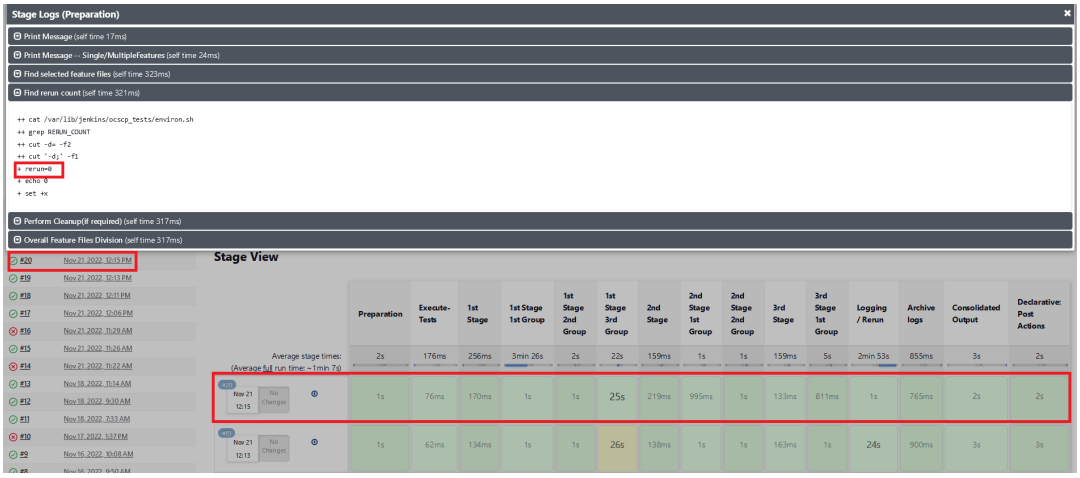
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases pass in the initial run. The pipeline will be green, and its status will show as blue.

Figure 2-34 Fetch_Log_Upon_Failure is set to YES and rerun_count is set to 0, test cases pass



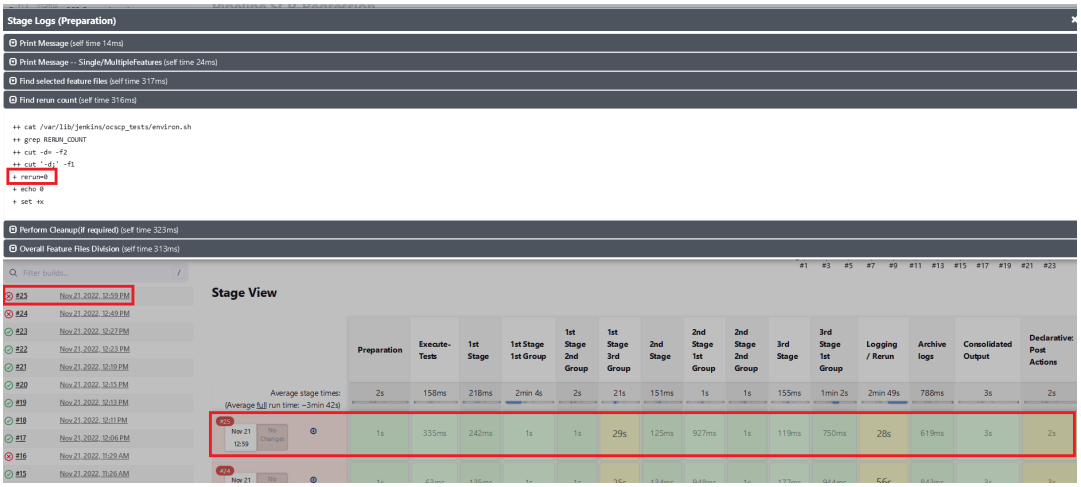
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases fail on the initial run but pass during the rerun. The initial execution stage is yellow and all subsequent successful stages will be green, and the status will be blue.

Figure 2-35 Test Cases Fail on the Initial Run but Pass in the Rerun



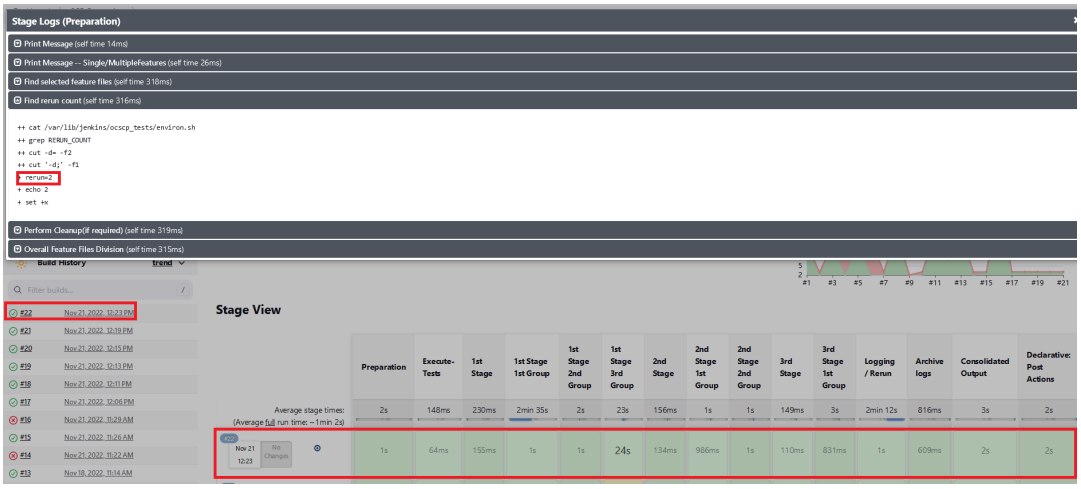
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases fail in both the initial and the rerun. Execution stages will show as yellow, all other successful stages will be shown as green, and the overall pipeline status will be red.

Figure 2-36 Test Cases Fail in Both the initial and the Rerun



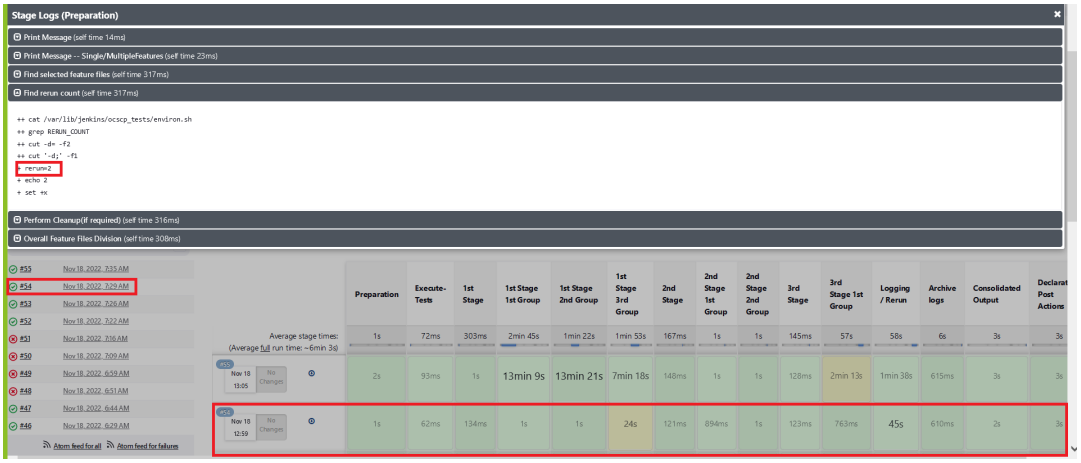
- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non-zero. If all of the test cases pass in the first run, no rerun will be initiated because the cases have already been passed. The pipeline will be green, and the status will be indicated in blue.

Figure 2-37 All of the Test cases Pass in the Initial Run



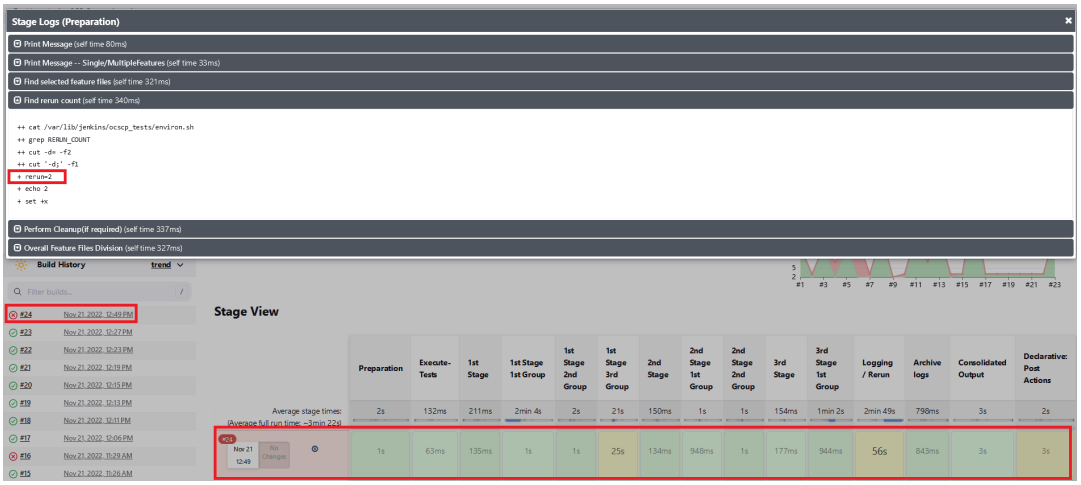
- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non-zero. If some of the test cases fail in the initial run and the remaining ones pass in one of the remaining reruns, then the initial test case execution stages will show as yellow, the remaining stages as green, and the overall pipeline status as blue.

Figure 2-38 Test Cases Fail in the Initial Run and the Remaining Ones Pass



- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non-zero. If some of the test cases fail in the initial run and the remaining ones fail in all the remaining reruns, the stages of test case execution will be shown in yellow, the remaining stages in green, and the overall pipeline status in red.

Figure 2-39 Test Cases Fail in the Initial and Remaining Reruns



- Whenever any of the multiple Behave processes that are running in the ATS are exited without completion, the stage in which the process exited and the consolidated output stage are shown as yellow, and the overall pipeline status will be yellow. Also in the consolidated output stage, near the respective stage result, the exact run in which the Behave processes exited without completion will be printed.

Figure 2-40 Stage View When Behave Process is Incomplete

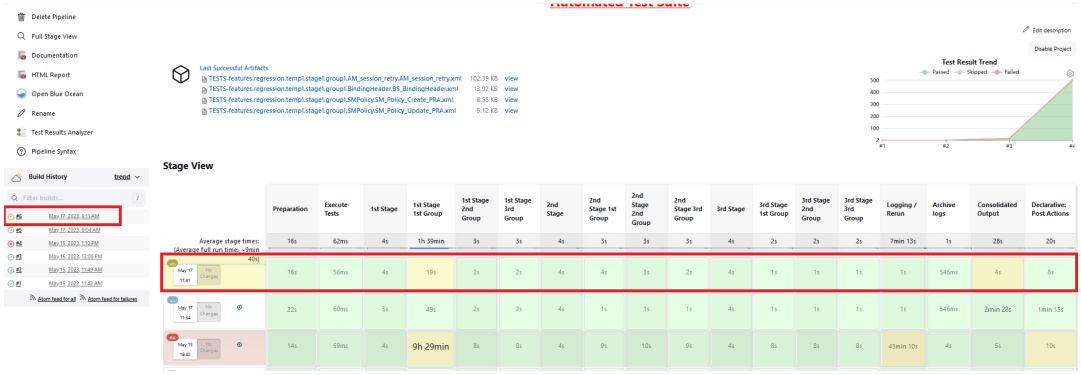
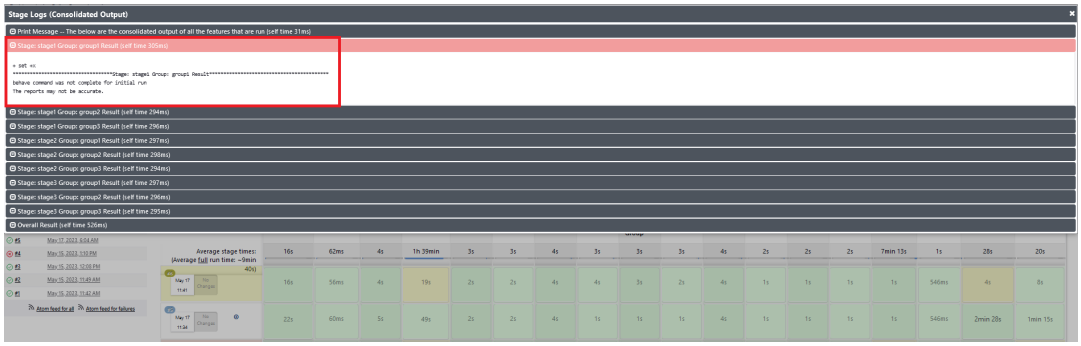


Figure 2-41 Consolidated Report for a Group When a Behave Process was Incomplete



Implementing Application Log

ATS automatically fetches the SUT Debug logs during the rerun cycle if it encounters any failures and saves them in the same location as the build console logs. The logs are fetched for the rerun time duration only using the timestamps. If, for some microservices, there are no log entries in that time duration, it does not capture them. Therefore, the logs are fetched only for the microservices that have an impact or are associated with the failed test cases.

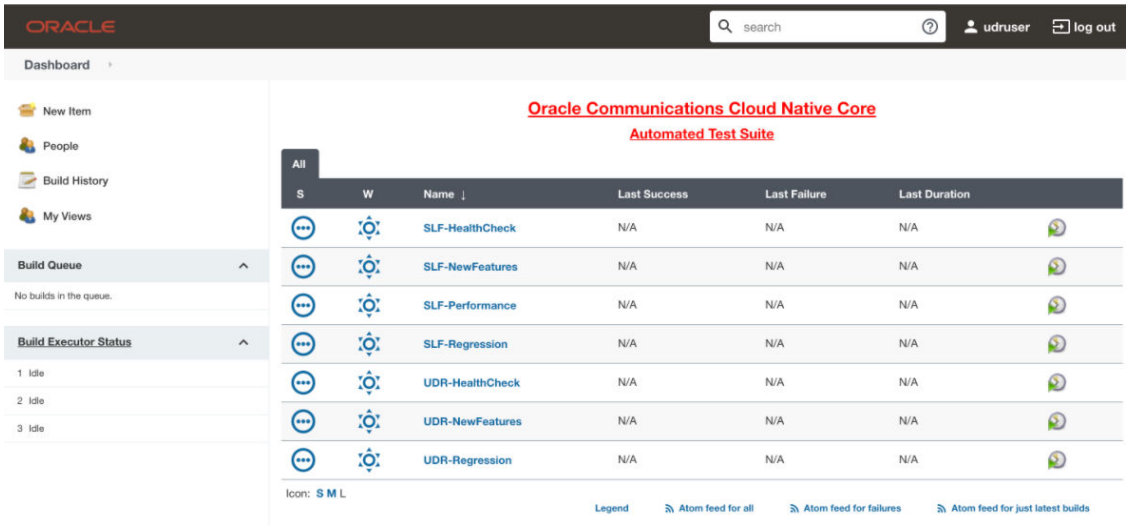
Location of SUT Logs: `/var/lib/jenkins/.jenkins/jobs/PARTICULAR-JOB-NAME/builds/BUILD-NUMBER/date-timestamp-BUILD-N.txt`

> **Note**
>
> The file name of the SUT log is added as a suffix with the date, timestamp, and build number (for which the logs are fetched). These logs share the same retention period as build console logs, set in the ATS configuration. It is recommended to set the retention period to optimal owing to the Persistent Volume Claim (PVC) storage space availability.
>
> .

2.11 Lightweight Performance

The Lightweight Performance feature allows you to run performance test cases. In ATS, a new pipeline known as "<NF>-Performance", where NF stands for Network Function, is introduced, for example, SLF-Performance.

Figure 2-42 Sample Screen: UDR Home Page



The <NF>-Performance pipeline verifies from 500 to 1k TPS (Transactions per Second) of traffic using the http-go tool, a tool used to run the traffic on the backend. It also helps to monitor the CPU and memory of microservices while running lightweight traffic.

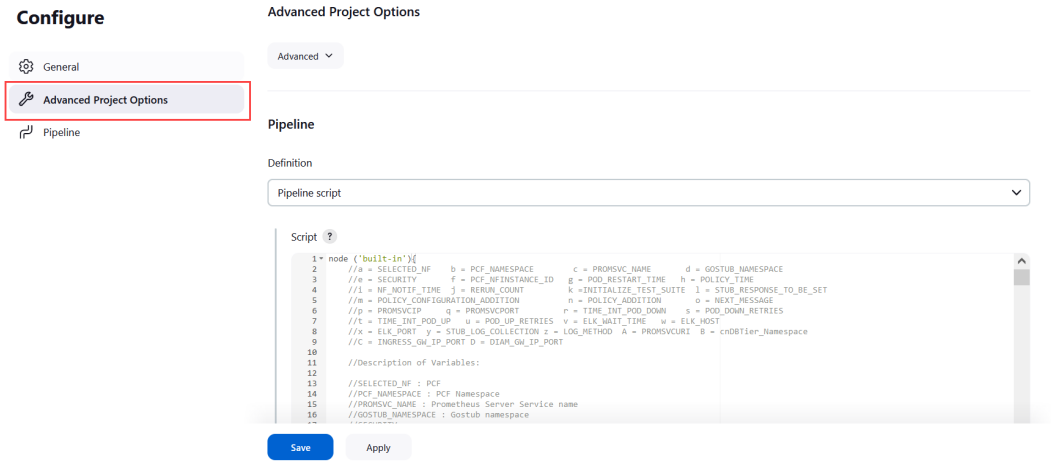
The duration of the traffic run can be configured on the pipeline.

2.11.1 Configure <NF>-Performance Pipeline

Perform the following to configure the performance pipeline:

1. On the NF home page, click **<NF>-Performance** pipeline, and then click **Configure**. The **General** tab appears. The user must wait for the page to load completely.
2. Click the **Advanced Project Options** tab. Scroll down to reach the **Pipeline** configuration section.

Figure 2-43 Advanced Project Options



3. Update the configurations as per your NF requirements and click **Save**. The **Pipeline <NF>-Performance** page appears.
4. Click **Build Now**. This triggers lightweight traffic for the respective network function. For more information, see #unique_65.

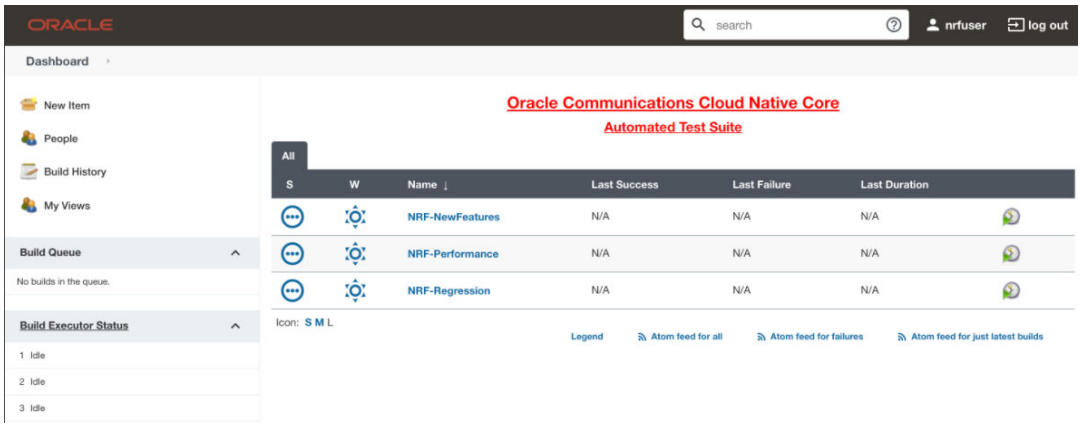
2.12 Modifying Login Password

You can log in to the ATS application using the default login credentials. The default login credentials are shared for each NF in the respective chapter of this guide.

Perform the following procedure to modify the default password:

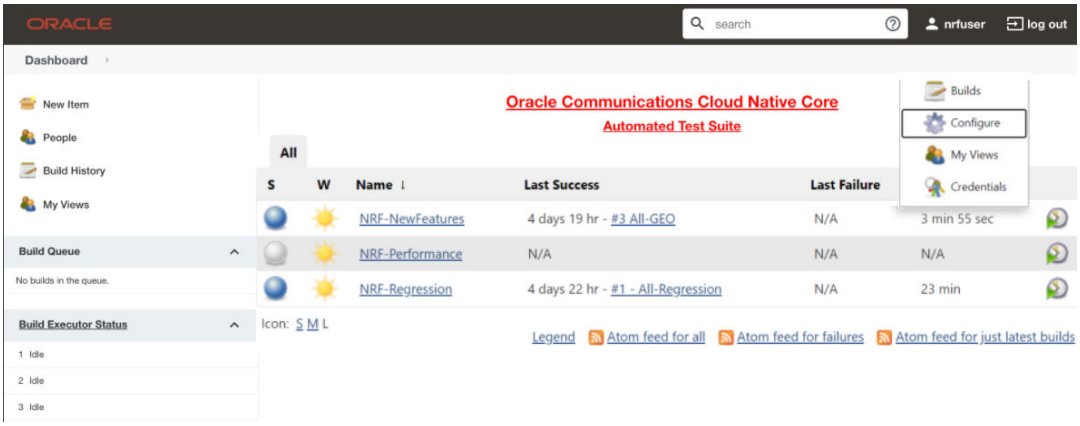
1. Log in to the ATS application using the default login credentials. The home page of the respective NF appears with its preconfigured pipelines as follows:

Figure 2-44 Sample Screen: NRF Home Page



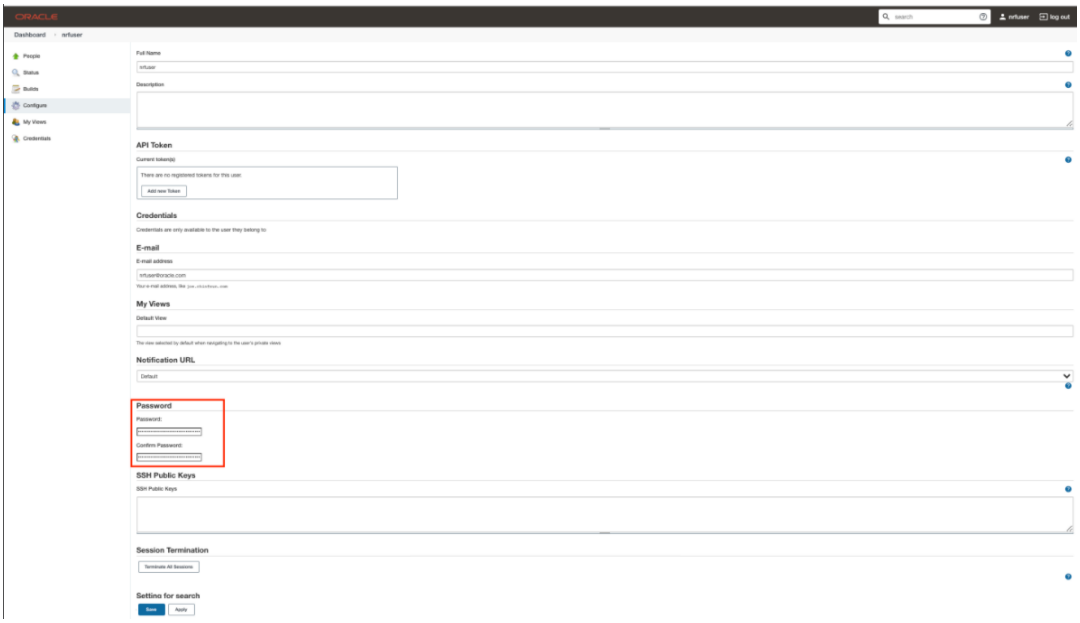
2. Hover over the user name and click the down arrow.
3. Click **Configure**.

Figure 2-45 Configure Option



The following page appears:

Figure 2-46 Logged-in User Details



4. In the Password section, enter the new password in the **Password** and **Confirm Password** fields.
5. Click **Save**.
   A new password is set for you.

2.13 Parallel Test Execution

Parallel test execution allows you to perform multiple logically grouped tests simultaneously on the same System Under Test (SUT) to reduce the overall execution time of ATS.

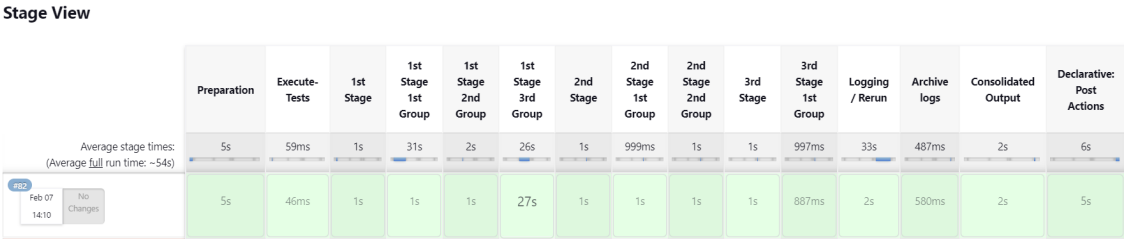
ATS currently runs all its tests in a sequential manner, which is time-consuming. With parallel test execution, tests can be run concurrently rather than sequentially or one at a time. Test cases or feature files are now separated into different folders, such as stages and groups, for concurrent test execution. Different stages, such as stage 1, stage 2, and stage 3, run the test

cases in a sequential order, and each stage has its own set of groups. Test cases or feature files available in different groups operate in parallel. When all the groups within one stage have completed their execution, only then the next stage will start the execution.

Pipeline Stage View

The pipeline stage view appears as follows:

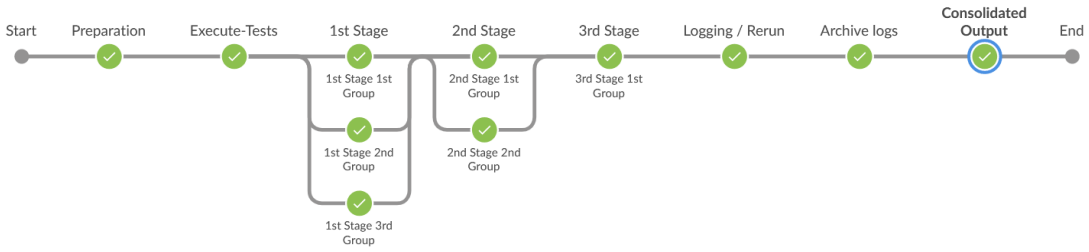
Figure 2-47 Pipeline Stage View



Pipeline Blue Ocean View

Blue Ocean is a Jenkins plugin that provides a better representation of concurrent execution with stages and groups. The pipeline blue ocean view appears as follows:

Figure 2-48 Pipeline Blue Ocean View



Impact on Other Framework Features

The integration of the parallel test framework feature has an impact on the following framework features. See the following sections for more details:

- Application Log Collection
- ATS API
- Managing Final Summary Report, Build Color, and Application Log
- PCAP Log Collection

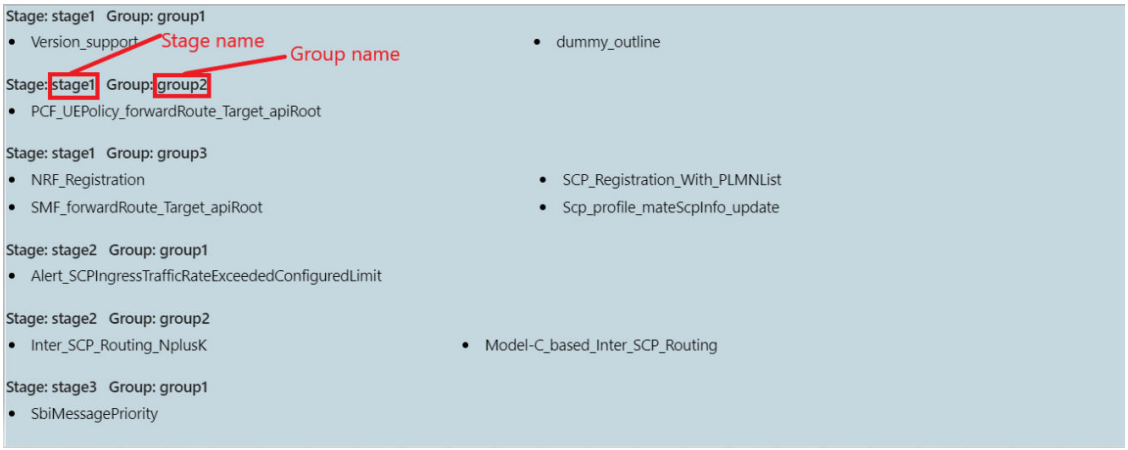
2.13.1 ATS GUI Page Changes

This section describes the changes to the ATS GUI page to trigger a build.

Changes in ATS GUI Page to Trigger a Build

The feature name, file name, and test case name are displayed under their stage and group names.

Figure 2-49 To Trigger a Build



2.13.2 ATS Console Log Changes

This section describes the changes to the ATS console log. The ATS console log contains logs of all the stages and groups. For more details, see Downloading or Viewing Individual Group Logs.

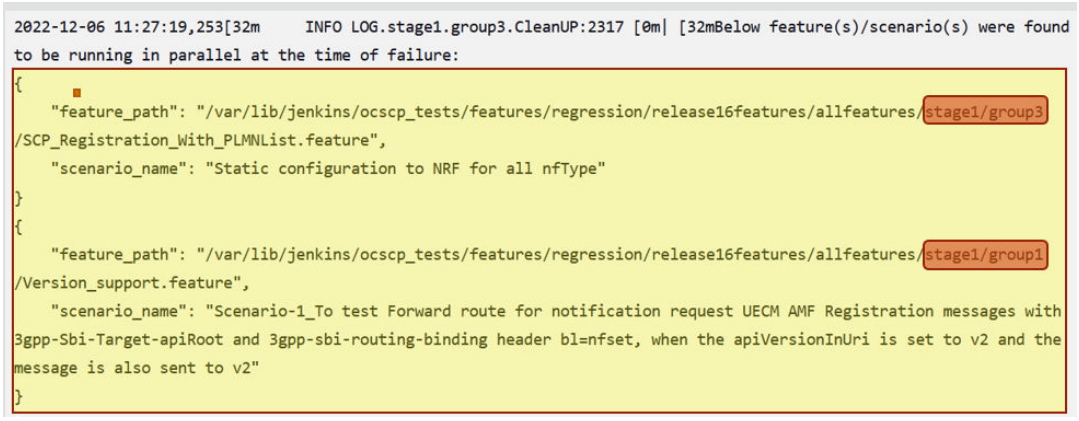
- A test case's stage and group names are listed in the logger statements for that test case.

Figure 2-50 Logger Statement

```
2022-12-06 09:34:53,762[32m     INFO LOG.stage2.group1.STEP:5453 [0m| [32m10.233.109.13[0m
2022-12-06 09:34:53,762[32m     INFO LOG.stage2.group1.STEP:5454 [0m| [32mudm1svc.scpsvc.svc.galaxy[0m
2022-12-06 09:34:53,893[32m     INFO LOG.stage2.group1.STEP:5459 [0m| [32m{"url":"/USEast/nudm-uecm/v1/imsi
```

- When a test case fails, a list of test cases running in parallel gets printed to make the debugging easier. The name of the test case and the absolute path to the feature file it belongs to are listed in this list.

Figure 2-51 Absolute Path of Feature File



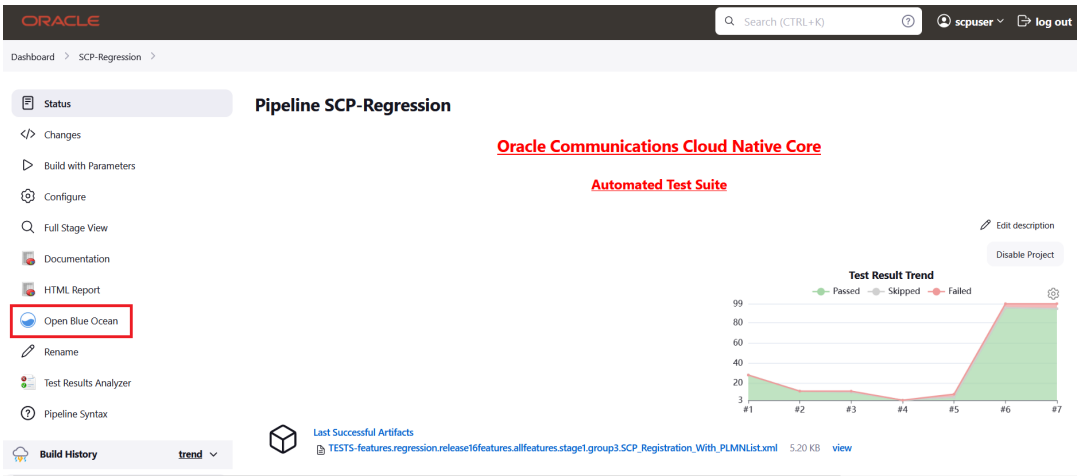
- The test result summary contains a summary for each group and an overall summary, along with the details of failing scenarios (stage-groupwise) and the total time taken by any pipeline execution. For further information, see the Final Summary Report, Build Colour Changes.

2.13.3 Downloading or Viewing Individual Group Logs

To download individual group logs:

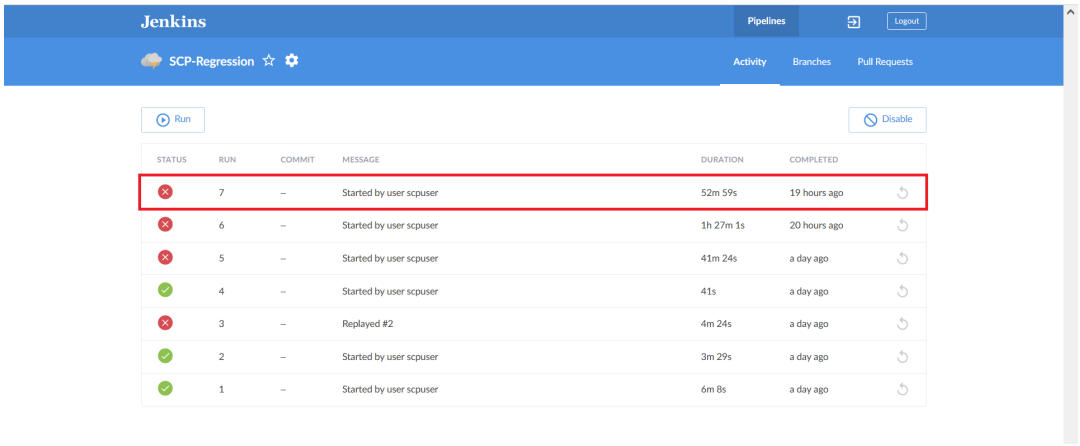
1. On the Jenkins pipeline page, click **Open Blue Ocean** in the left navigation pane.

Figure 2-52 Jenkins Pipeline Page



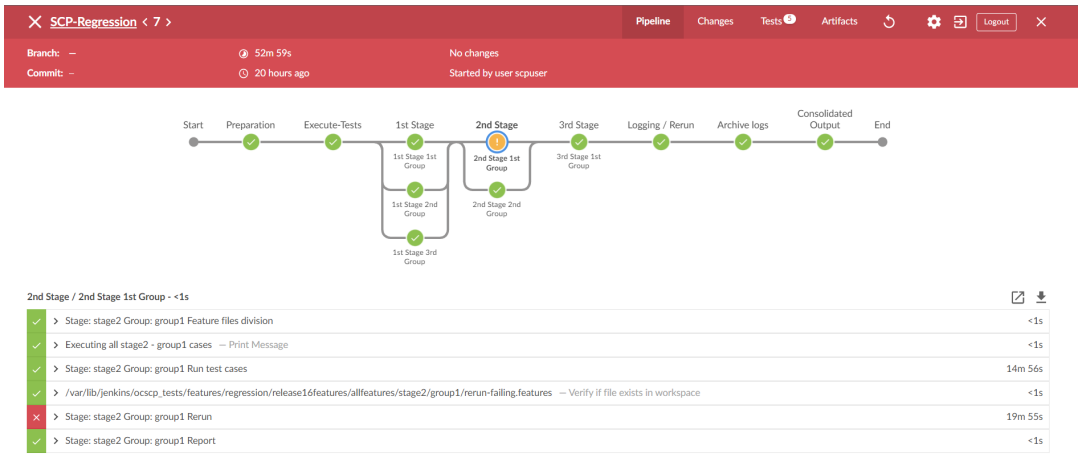
2. Click the desired build row on the Blue Ocean page.

Figure 2-53 Run the Build



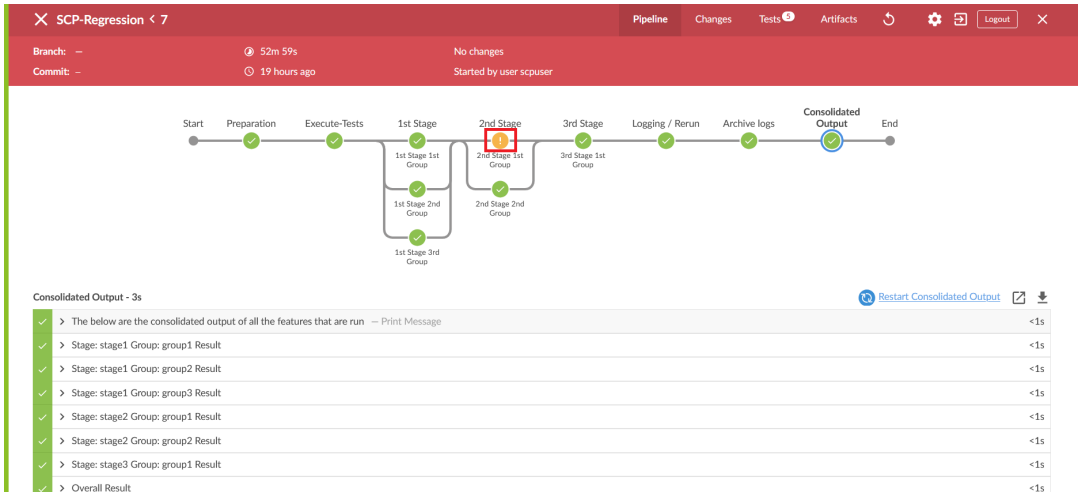
3. The selected build appears. The diagram displays the order in which the different stages, or groups, are executed.

Figure 2-54 Stage Execution



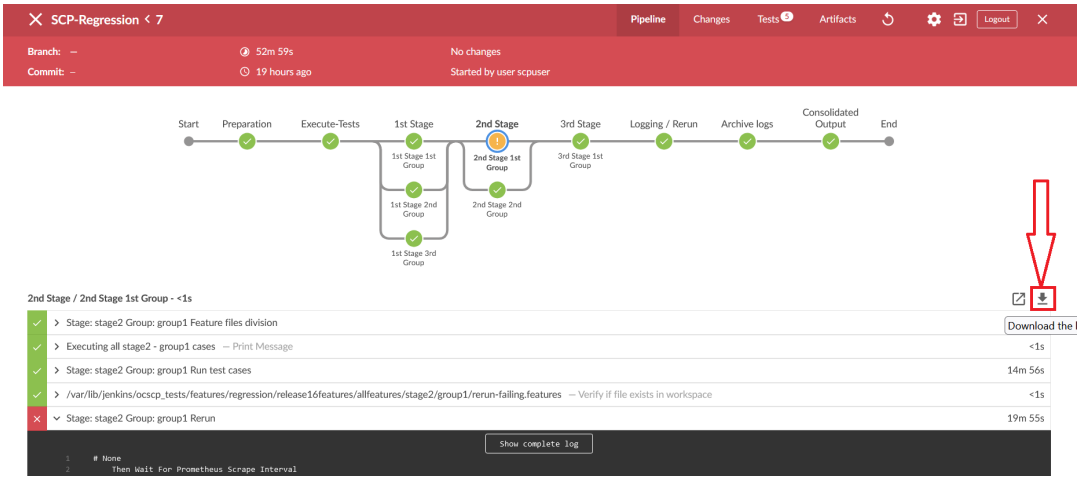
4. Click the desired group to download the logs.

Figure 2-55 Executed Groups



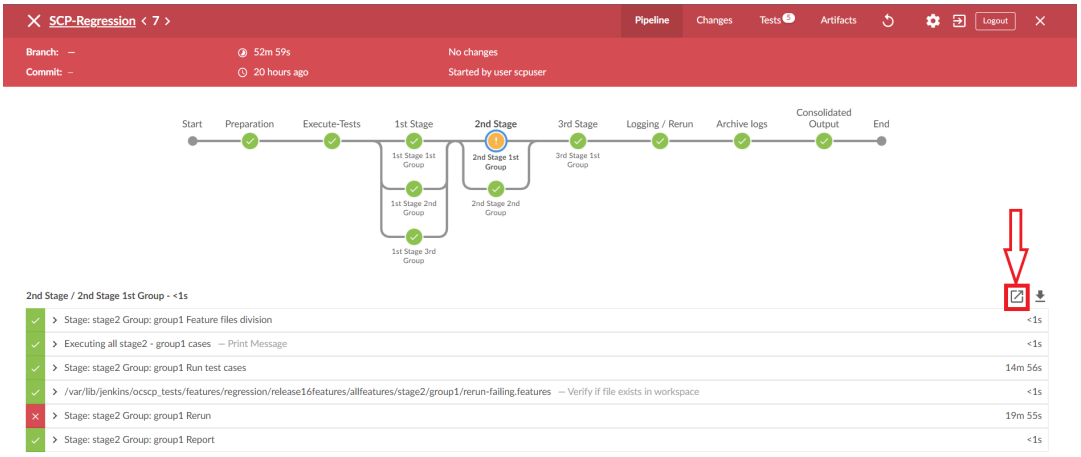
5. Click the **Download** icon on the bottom right of the pipeline. The log for the selected group is downloaded to the local system.

Figure 2-56 Download Logs



6. To view the log, click the **Display Log** icon. The logs are displayed in a new window.

Figure 2-57 Display Logs



Viewing Individual Group Logs without using Blue Ocean

There are two alternate ways to view individual group logs:

- Using Stage View
  - On the Jenkins pipeline page, hover the cursor over the group in stage view to view the logs.
  - A pop-up with the label "Logs" will appear. Click on it.
  - There will be a new pop-up window.It contains many rows, where each row corresponds to the execution of one Jenkins step.
  - Click on the row labelled **Stage: stage_name>."Group: <group_name> Run test cases** to view the log for this group's execution.

  - Click on the row labelled **Stage: stage_name>." "group_name> Rerun** to display the re-run logs.
- Using Pipeline Steps Page
  - On the Jenkins pipeline page, under the **Build History** dropdown, click on the desired build number.
  - Click the **Pipeline Steps** button on the left pane.
  - A table with columns for step, arguments, and status appears.
  - Under the Arguments column, find the label for the desired stage and group.
  - Click on the step with the label **Stage: <stage_name> Group: <group_name> Run test** cases under it or click the **Console output** icon near the status to view the log for this group execution.
  - To see rerun logs, find the step with the label **Stage: <stage_name> Group: <group_name> Rerun** under it.

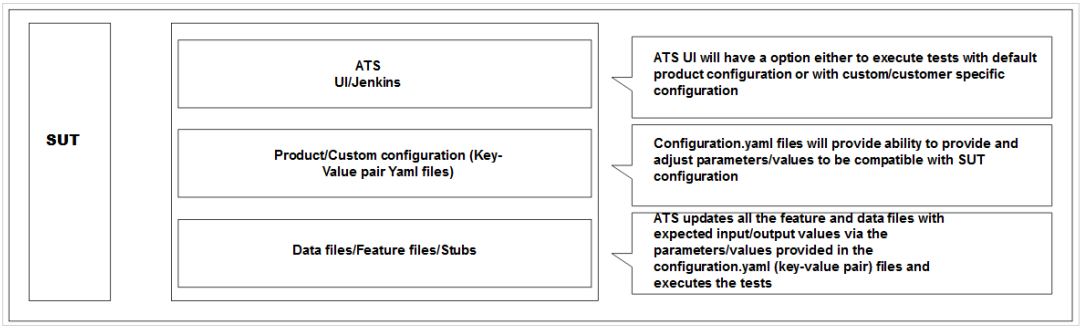
2.14 Parameterization

This feature allows you to provide or adjust values for the input and output parameters needed for the test cases to be compatible with the SUT configuration. You can update or adjust the key-value pair values in the `global.yaml` and `feature.yaml` files for each of the feature files so that they are compatible with SUT configuration. In addition to the existing custom test case folders (Cust New Features, Cust Regression, and Cust Performance), this feature enables folders to accommodate custom data, default product configuration, and custom configuration. You can maintain multiple versions or copies of the custom data folder to suit varied or custom SUT configurations. With this feature, the ATS GUI has the option to either execute test cases with the default product configuration or with a custom configuration.

This feature enables you to perform the following tasks:

- Define parameters and assign or adjust values to make them compatible with SUT configuration.
- Execute test cases either with default product configurations or custom configurations and multiple custom configurations to match varied SUT configurations.
- Assign or adjust values for input or output parameters through custom or default configuration yaml files (key-value pair files).
- Define or adjust the input or output parameters for each feature file with its corresponding configuration.
- Create and maintain multiple configuration files to match multiple SUT configurations.

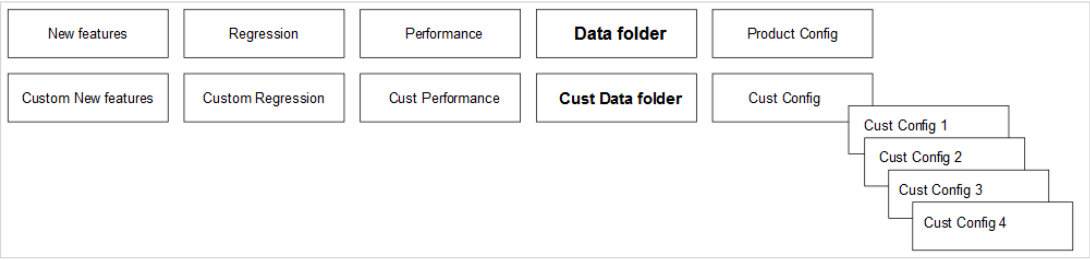
Figure 2-58 SUT Design Summary



In the folder structure:

- The Product Config folder contains default product configuration files (feature-wise yaml per key-value pair), which are compatible with default product configuration.
- New features, Regression and Performance, Data folder, and Product Config folders are replicated or copied into custom folders and delivered as part of the ATS package in every release.
- You can customize custom folders by:
  - Removing test cases not needed or as appropriate for your use.
  - Adding new test cases as needed or as appropriate for your use.
  - Removing or adding data files in the cust_data folder or as appropriate for your use.
  - Adjusting the parameters or values in the key-value pair per yaml file in the custom config folder for test cases to run or pass with a custom configured SUT.
- The product folders are always intact (unchanged) and you can update the Custom folders
- You can maintain multiple copies of Custom Configurations and bring them to use as needed or as appropriate for the SUT configuration.

Figure 2-59 Folder Structure



2.14.1 Running Test Cases

Enable

Rename or copy the Cust Config [1/2/3/N] folder to the Cust Config folder in order for ATS to run the test cases with a specific custom configuration. When the option to run test cases with custom configuration is chosen, it always points to the Cust Config folder.

To Run ATS Test Cases

ATS has the option to run test cases with default or custom configuration.

- If custom configuration is selected, then test cases from custom folders are populated on the ATS UI, and custom configuration is applied to them through the key-value pair per yaml file defined or present in the "Cust Config" folder.
- If product configuration is selected, then the test cases from product folders are populated on the ATS UI, and product configuration is applied to them through key-value pairs per yaml file defined or present in the Product Config folder.

Figure 2-60 ATS Execution Flow

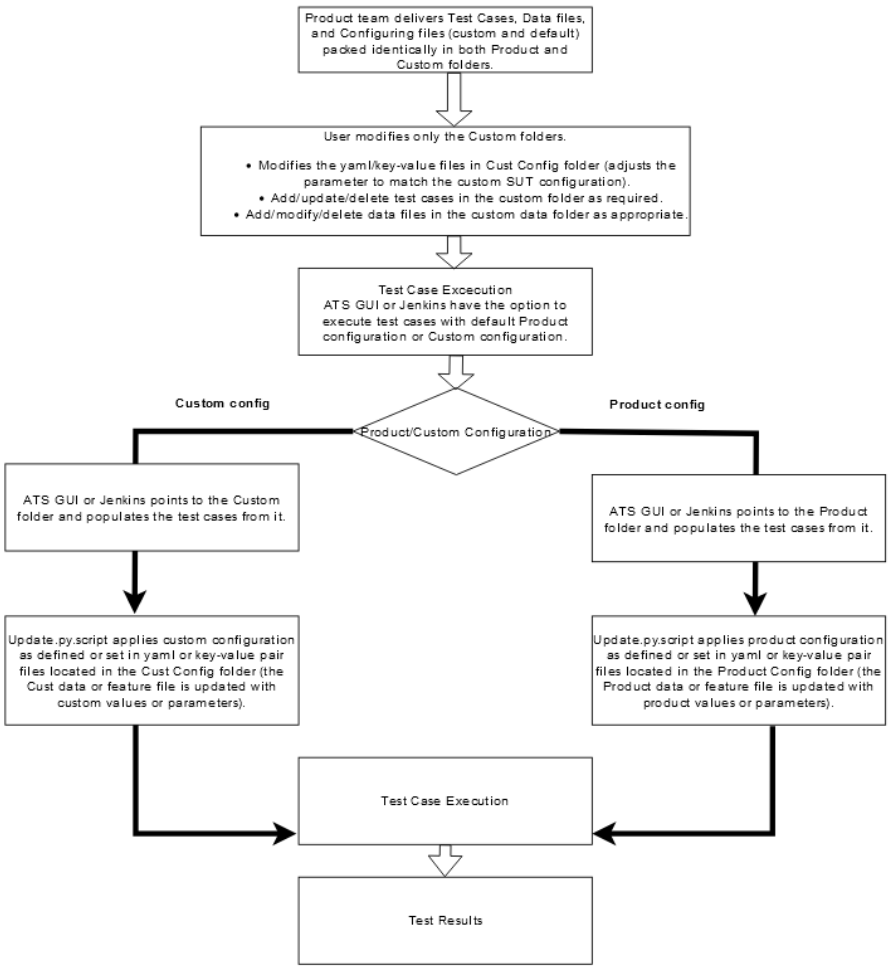
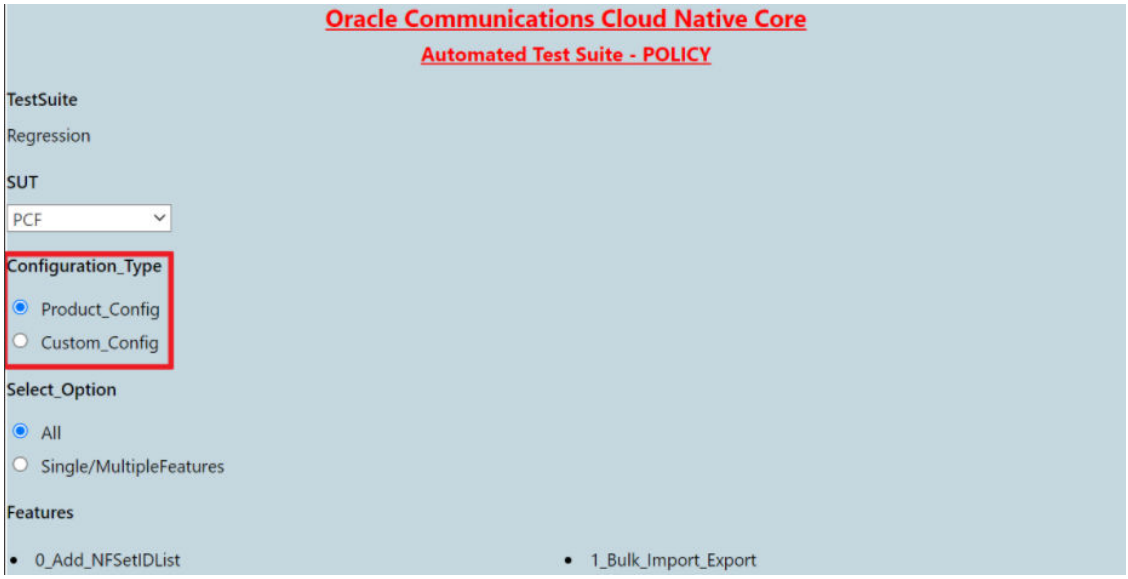


Figure 2-61 Sample: Configuration_Type



2.15 PCAP Log Collection

PCAP Log Collection allows collecting the NF, SUT, or PCAP logs from the debug tool sidecar container. This feature can be integrated and delivered as a standalone or along with the Application Log Collection feature. For information, see Application Log Collection.

PCAP Log Integration

1. The Debug tool should be enabled on SUT Pods while deploying the NF. The name of the Debug container must be "tools".
   For example, in SCP, the debug tool should be enabled for all the SCP microservice pods.
2. Update the following parameters in the values.yaml file, under the resource section, with ATS minimum resource requirements:
   a. CPU: 3
   b. memory: 3Gi
3. On the home page, click any new feature or regression pipeline.
4. In the left navigation pane, click **Build with Parameters**.
5. Select YES from the drop-down menu of **Fetch_Log_Upon_Failure**.
6. If option **Log_Type** is available, select value **PcapLog [Debug Container Should be Running]** for it.
7. Select **PcapLog [Debug Container Should be Running]** to activate PCAP Log Collection in ATS-NF.
   The following Build with Parameters page appears when only the PCAP logs feature has been integrated.

Figure 2-62 PCAP Logs Selection Option

8. After the build execution is complete, go into the ATS pod, then navigate to below path to find the pcaplogs: `.jenkins/jobs/<Pipeline Name>/builds/<build number>/` For example, `.jenkins/jobs/SCP-Regression/builds/5/`

   Pcaplogs is present in zip form. Unzip it to get the log files.

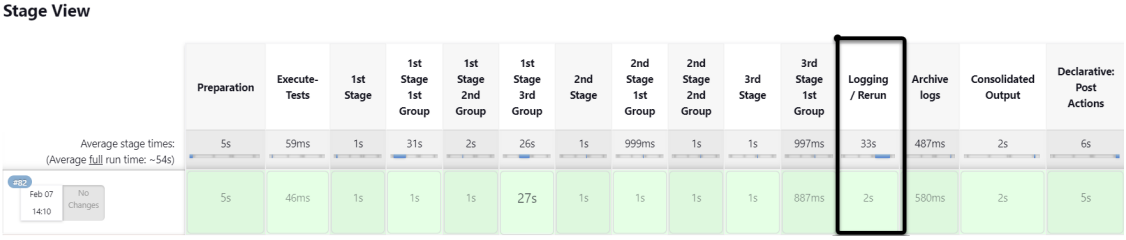
The following Build with Parameters page appears when both application and PCAP logs have been integrated.

Figure 2-63 Both Application Logs and PCAP Logs Selection

2.15.1 Application Log Collection and Parallel Test Execution Integration

A new stage,"Logging/Rerun", has been added at the end of the Execute-Tests stage to collect rerun logs, such as applog and PCAP logs, by running the failed test cases in a sequence.

Figure 2-64 Logging/Rerun new stage



If the `Fetch_Log_Upon_Failure` parameter is set to YES and if any test case fails in the initial run, then:

- The failed test case reruns and log collection start in the Logging/Rerun stage after the initial run is completed for all the test cases.
- The logs from the initial execution are collected, but they might be incorrect.
- Even if the `rerun` parameter is set to 0, the failed test case reruns in the Logging/Rerun stage and the log is collected.
- If the `Fetch_Log_Upon_Failure` parameter is set to NO and if any test case fails in the initial run, then the failed test case rerun starts in the same stage after the initial execution is over for all the test cases in its group.

2.16 Persistent Volume for 5G ATS

The Persistent Volume (PV) feature allows ATS to retain historical build execution data, test cases, and ATS environment configurations.

ATS Packaging When Using Persistent Volume

- **Without the Persistent Volume option:** ATS package includes an ATS image with test cases.
- **With Persistent Volume option:** ATS package includes the ATS image and test cases separately. The new test cases are provided between the releases.
  To support both with and without Persistent Volume options, test cases and execution job data are packaged in the ATS image as well as a tar file.

2.16.1 Processing Flow

First Time Deployment

Initially, when you deploy ATS, for example, the PI-A ATS pod, you use PVC-A, which is provisioned and mounted to the PI-A ATS pod. By default, PVC-A is empty. So, you have to copy the data (ocslf_tests and jobs folders) from the PI-A tar file to the pod after the pod is up and running. Then restart the PI-A pod. At this point, you can change the number of build logs to maintain in the ATS GUI.

Subsequent Deployments

When you deploy ATS for the subsequent time, for example, in a PI-B ATS pod, you use PVC-B, which is provisioned and mounted to the PI-B ATS pod. By default, the PVC-B is empty, and you have to copy the data (ocslf_tests and jobs folders) from the PI-B tar file to the pod after the pod is up and running. At this point, copy all the necessary changes to the PI-B pod from the PI-A pod and restart the PI-B pod. You can change the number of build logs to maintain in

the ATS GUI. After updating the number of builds, you can delete the PI-A pod and continue to retain the PVC-A. If you do not want backward porting, you can delete PVC-A.

2.16.2 Deploying Persistent Volume

Preinstallation Steps

1. Before deploying Persistent Volume, create a PVC in the same namespace where you have deployed ATS. You have to provide values for the following parameters to create a PVC:
   - PVC Name
   - Namespace
   - Storage Class Name
   - Size of the PV
2. Run the following command to create a PVC:

```
kubectl apply -f - <<EOF

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: <Please Provide the PVC Name>
  namespace: <Please Provide the namespace>
  annotations:
spec:
  storageClassName: <Please Provide the Storage Class Name>
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <Please Provide the size of the PV>
EOF
```

> **Note**
>
> It is recommended to suffix the PVC name with the release version to avoid confusion during subsequent releases. For example: ocats-slf-pvc-1.9.0

3. The output of the above command with parameters is as follows:

```
[cloud-user@atscne-bastion-1 templates]$ kubectl apply -f - <<EOF
>
> apiVersion: v1
> kind: PersistentVolumeClaim
> metadata:
>     name: ocats-slf-1.9.0-pvc
>     namespace: ocslf
>     annotations:
> spec:
>     storageClassName: standard
>     accessModes:
```

```
>         - ReadWriteOnce
>     resources:
>         requests:
>             storage: 1Gi
> EOF
```

The persistentvolumeclaim/ocats-slf-1.9.0-pvc is created.

4. To verify whether PVC is bound to PV and is available for use, run the following command:

   ```
   kubectl get pvc -n <namespace used for pvc creation>
   ```

   The output of the above command is as follows:

Figure 2-65 Verifying PVC

```
[cloud-user@atscne-bastion-1 templates]$ kubectl get pvc -n ocslf
NAME                  STATUS   VOLUME                                     CAPACITY   ACCESS MODES STORAGECLASS   AGE
ocats-slf-1.9.0-pvc   Bound    pvc-1ee50daf-2eb7-4289-b541-51c111270513   1Gi        RWO          standard       8s
policy-pvc-test       Bound    pvc-d346d5c4-129e-4f13-8bd0-a7b7555e0e2f   1Gi        RWO          standard       31d
slf-pvc               Bound    pvc-0b0bce8c-b856-4e0e-b4d8-0f981250adbd   1Gi        RWO          standard       12d
```

Check that the STATUS is **Bound** and that the rest of the parameters, such as NAME, CAPACITY, ACCESS MODES, STORAGECLASS, and so on, are the same as specified in the PVC creation command.

> **Note**
>
> Do not proceed further if there is any issue with PVC creation. Contact your administrator to create a PV.

5. After creating persistent volume, change the following parameters in the values.yaml file to deploy persistent volume.
   - Set the **PVEnabled** parameter to "true".
   - Provide the value for the **PVClaimName** parameter. The PVClaimName value should be the same as the value used to create a PVC.

Postinstallation Steps

1. After deploying ATS, copy the <nf_main_folder> and <jenkins jobs> folders from the tar file to their ATS pod, and then restart the pod as a one-time activity.
   a. Run the following command to extract the tar file:

      ```
      ocats-<nf_name>-data-<release-number>.tgz
      ```

> **Note**
>
> The ats_data.tar file is the name of the tar file containing <nf_main_folder> and jobs folders. It can be different for different NFs.

b. Run the following set of commands to copy the required folders:

```
kubectl cp ats_data/jobs <namespace>/<pod-name>:/var/lib/
jenkins/.jenkins/
```

```
kubectl cp ats_data/<nf_main_folder> <namespace>/<pod-name>:/var/lib/
jenkins/
```

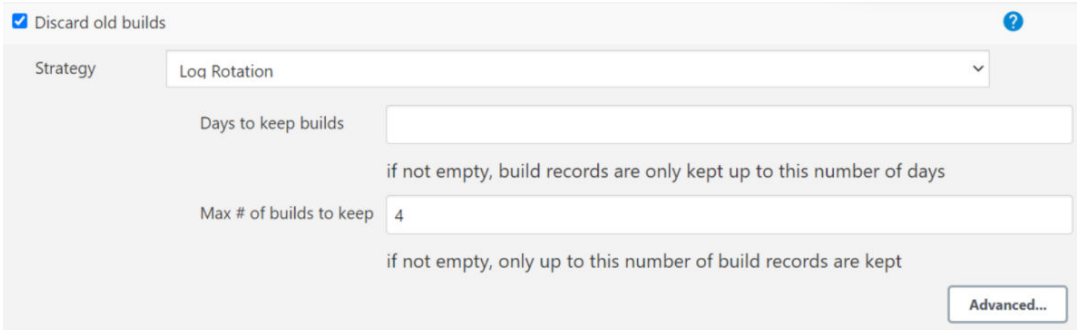
c. Run the following command to restart the pods as one-time activity:

```
kubectl delete po <pod-name> -n <namespace>
```

> **Note**
>
> Before running the following command, copy the changes done on the new release pod from the old release pod using the `kubectl cp` command. **[Applicable in the case of subsequent deployment only]**

2. When the pod is up and running, log in to the ATS GUI and go to your NF specific pipeline. Click **Configure** in the left navigation pane. The **General** tab appears. Configure the **Discard old Builds** option. This option allows you to configure the number of builds you want to retain in the persistent volume.

Figure 2-66 Discard Old Builds



> **Note**
>
> It is recommended to configure this option. If you do not input a value for this option, the application will take into account all builds, which could be a large number, and will completely consume the Persistent Volume.

2.16.3 Backward Porting

The following deployment steps apply to the old release of PVC-supported ATS Pod.

Prerequisite: You should have the old PVC that contains the old release of POD data.

> **Note**
>
> This procedure is for backward porting purposes only and should not be considered a subsequent release of the POD deployment procedure.

The deployment procedure for the old release PVC-supported ATS pod is the same; however, while deploying the ATS pod, you have to update the values.yaml file with the following:

- Change the **PVEnabled** parameter to "true".
- Provide the name of the old PVC as the value for parameter **PVClaimName**.

2.17 Test Results Analyzer

The Test Results Analyzer is a plugin available in ATS to view pipeline test results based on XML reports. It provides the test results report in a graphical format, which includes consolidated and detailed stack trace results in case of any failures. It allows you to navigate to each and every test.

The test result report shows any one of the following statuses for each test case:

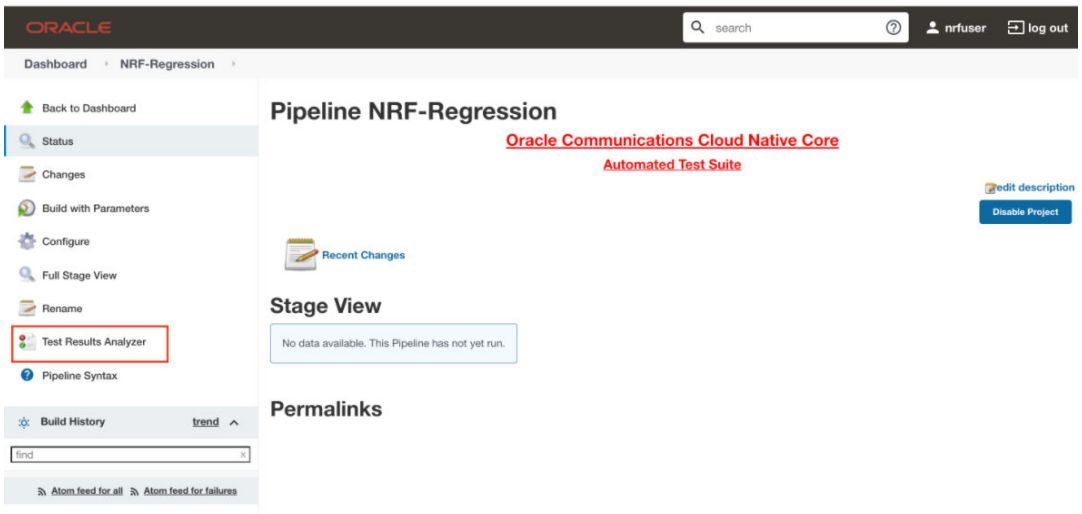
- **PASSED:** If the test case passes.
- **FAILED:** If the test case fails.
- **SKIPPED:** If the test case is skipped.
- **N/A:** If the test cases are not executed in the current build.

2.17.1 Accessing Test Results Analyzer Feature

To access the test results analyzer feature:

1. From the NF home page, click any new feature pipeline or regression pipeline where you want to run this plugin.
2. In the left navigation pane, click **Test Results Analyzer**.

Figure 2-67 Test Results Analyzer Option



When the build completes, the test result report appears. A sample test result report is shown below:

Figure 2-68 Sample Test Result Report

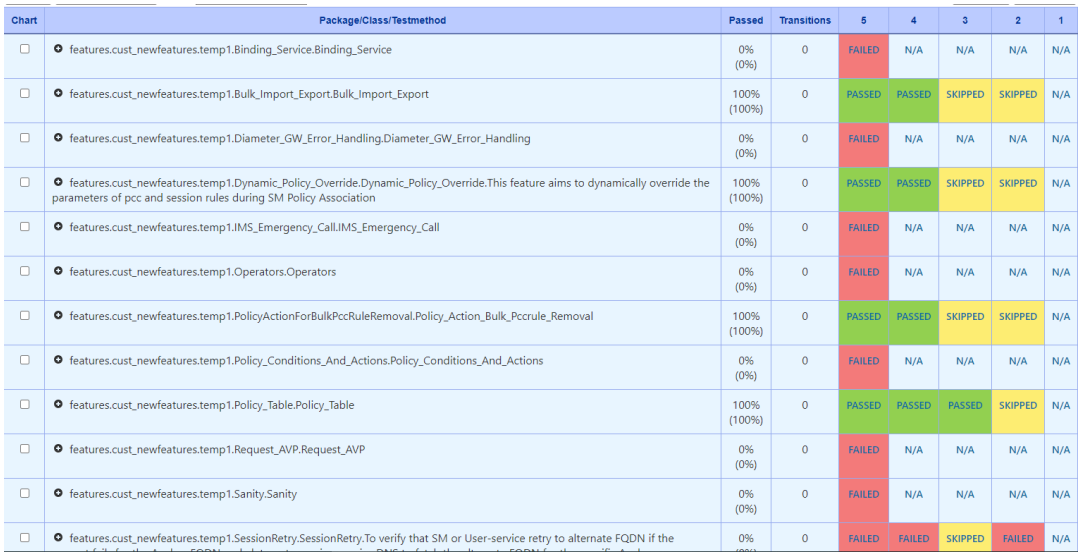
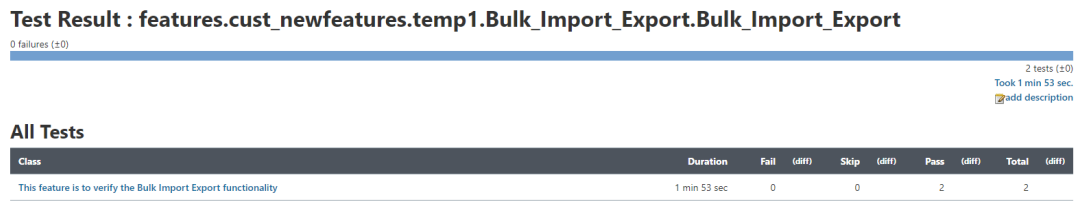


Chart	Package/Class/Testmethod	Passed	Transitions	5	4	3	2	1
☐	features.cust_newfeatures.temp1.Binding_Service.Binding_Service	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Bulk_Import_Export.Bulk_Import_Export	100% (100%)	0	PASSED	PASSED	SKIPPED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.Diameter_GW_Error_Handling.Diameter_GW_Error_Handling	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Dynamic_Policy_Override.Dynamic_Policy_Override.This feature aims to dynamically override the parameters of pcc and session rules during SM Policy Association	100% (100%)	0	PASSED	PASSED	SKIPPED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.IMS_Emergency_Call.IMS_Emergency_Call	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Operators.Operators	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.PolicyActionForBulkPccRuleRemoval.Policy_Action_Bulk_Pccrule_Removal	100% (100%)	0	PASSED	PASSED	SKIPPED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.Policy_Conditions_And_Actions.Policy_Conditions_And_Actions	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Policy_Table.Policy_Table	100% (100%)	0	PASSED	PASSED	PASSED	SKIPPED	N/A
☐	features.cust_newfeatures.temp1.Request_AVP.Request_AVP	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.Sanity.Sanity	0% (0%)	0	FAILED	N/A	N/A	N/A	N/A
☐	features.cust_newfeatures.temp1.SessionRetry.SessionRetry.To verify that SM or User-service retry to alternate FQDN if the	0%	0	FAILED	FAILED	SKIPPED	FAILED	N/A

3. Click any one of the statuses (PASSED, FAILED, SKIPPED) to view the respective feature detail status report.

> **Note**
>
> For N/A status, a detailed status report is not available.

Figure 2-69 Test Result



Test Result : features.cust_newfeatures.temp1.Bulk_Import_Export.Bulk_Import_Export

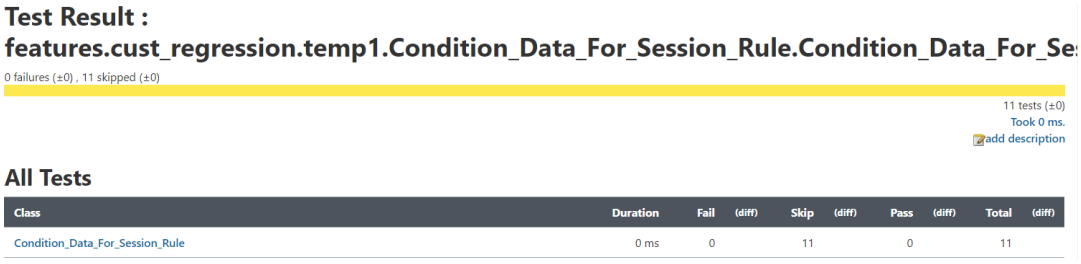
0 failures (±0)

2 tests (±0)
Took 1 min 53 sec.
add description

All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
This feature is to verify the Bulk Import Export functionality	1 min 53 sec	0		0		2		2	

Figure 2-70 Test Result



Test Result : features.cust_regression.temp1.Condition_Data_For_Session_Rule.Condition_Data_For_Se

0 failures (±0) , 11 skipped (±0)

11 tests (±0)
Took 0 ms.
add description

All Tests

Class	Duration	Fail	(diff)	Skip	(diff)	Pass	(diff)	Total	(diff)
Condition_Data_For_Session_Rule	0 ms	0		11		0		11	

4. In the case of a rerun, the test cases that passed in the initial run but were skipped in the rerun are considered "passed" in the Test Results Analyzer Report. The following screenshot depicts the scenario: "Variant2_equal_smPolicySnssaiData,Variant2_exist_smPolicyData,Variant2_exist_smPolicyDnnData_dnn" where the test cases passed in the initial run but skipped in the rerun are considered "passed" in general.

Figure 2-71 Test Results

5. Click **PASSED**. The following highlighted message means the test case was passed in the main run but skipped in the rerun.

Figure 2-72 Test Result Info

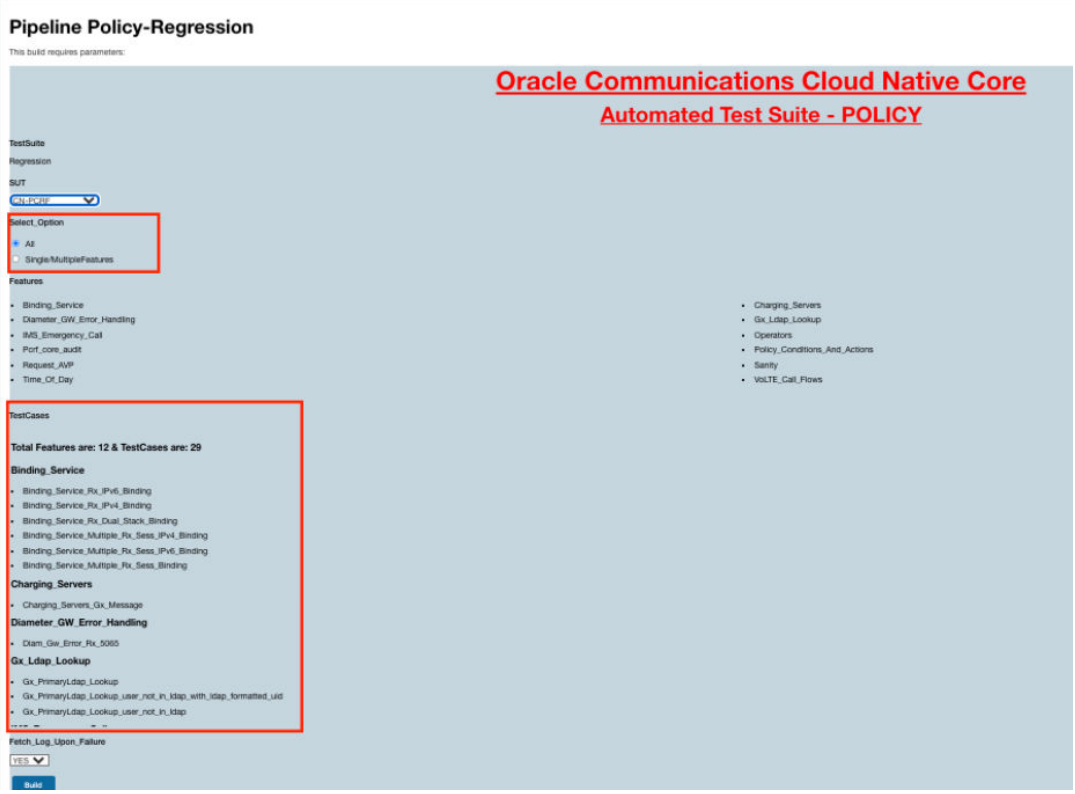
2.18 Support for Test Case Mapping and Count

The Test Case Mapping and Count feature displays the total number of features, test cases, or scenarios and their mapping to each feature in the ATS GUI.

2.18.1 Access Test Case Mapping and Count Feature

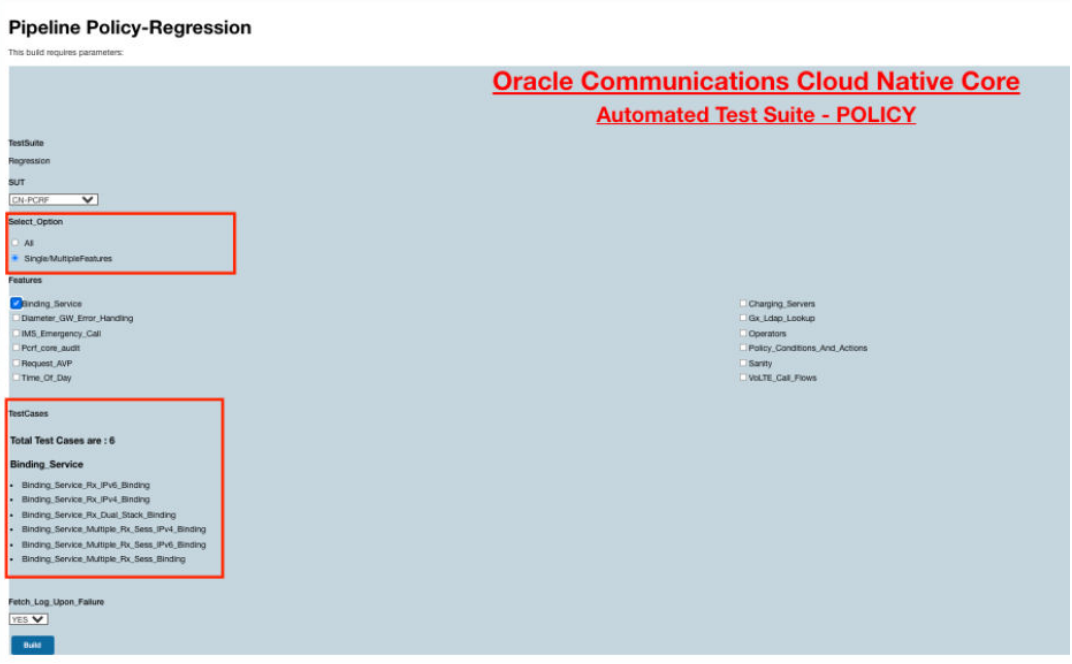
1. On the NF home page, click any new feature or regression pipeline where you want to use this feature.
2. In the left navigation pane, click **Build with Parameters**.
3. Select **All** from the Select_Option to view the TestCases details mapped to each feature.

Figure 2-73 Test Case Mapping



4. Select **Single/MultipleFeatures** from the Select_Option to view the the test cases details.

Figure 2-74 Test Cases Details When Select_Option is Single/MultipleFeatures



2.19 Support for Transport Layer Security

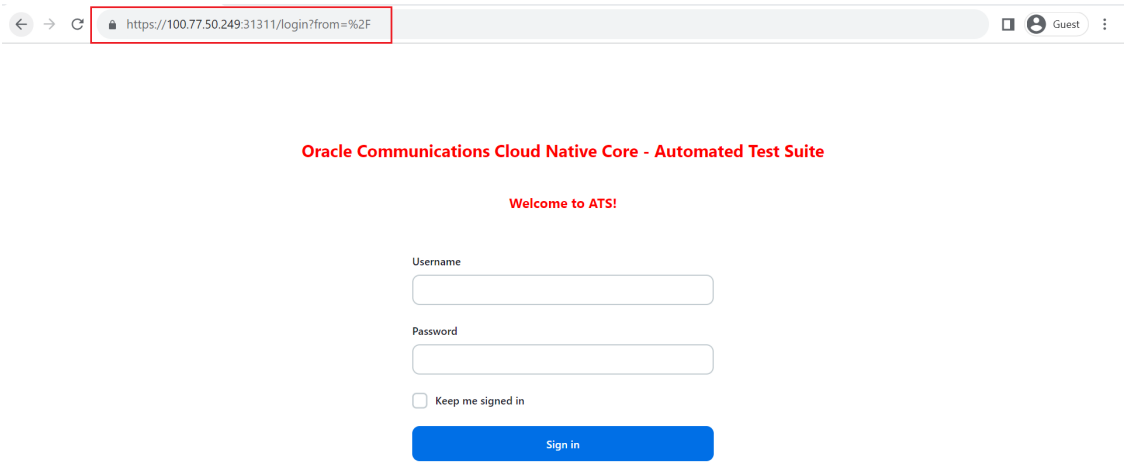
Currently, ATS is accessible through HTTP, which can raise security risks.

With the support of the TLS feature, Jenkins servers have been upgraded to support HTTPS, ensuring a secure and encrypted connection when accessing the ATS dashboard.

To provide encryption, HTTPS uses an encryption protocol known as Transport Layer Security (TLS), which is a widely accepted standard protocol that provides authentication, privacy, and data integrity between two communicating computer applications.

Now, users can access the ATS GUI with the HTTPS protocol instead of the previously used HTTP protocol.

Figure 2-75 Access with TLS



> **Note**
>
> If this feature is not enabled before installation, HTTP will continue to operate.

3 Installing ATS for Different Network Analytics Suite Products

This section describes how to install ATS for different Network Analytics Suite products. It includes:

- Installing ATS for NWDAF
- Installing ATS for OCNADD

3.1 Installing ATS for NWDAF

This section describes the resource requirements and ATS installation procedures for NWDAF:

- Software Requirements
- Environment Setup
- Resource Requirements
- Downloading the ATS Package
- Pushing the Images to Customer Docker Registry
- Configuring ATS
- Deploying NWDAF ATS in the Kubernetes Cluster
- Verifying ATS Deployment
- Creating and Verifying NWDAF Console Namespaces

3.1.1 Software Requirements

This section describes the software requirements to install ATS for NWDAF. Install the following software bearing the versions mentioned in the table below:

Table 3-1 Software Requirements

Software	Version
Kubernetes	1.20.7, 1.21.7, 1.22.5,1.24.6,1.25.6, 1.26.5,1.27.5
Helm	3.1.2, 3.5.0, 3.6.3, 3.8.0, 3.9.4, 3.10.3,3.12.0,3.12.3
Podman	2.2.1, 3.2.3, 3.3.1, 4.4.1, 4.2.0, 4.6.1

Supported CNE versions are: Release 1.9.x ,1.10.x, 22.1.x and 23.1.x, 23.2.x, 23.3.x, and 23.4.x.

To verify the CNE version, run the following command:

```
echo $OCCNE_VERSION
```

To verify the Helm and Kubernetes versions installed in the CNE, run the following commands:

- Verify Kubernetes version:

  ```
  kubectl version
  ```

- Verify Helm version:

  ```
  helm3 version
  ```

3.1.2 Environment Setup

This section describes steps to ensure the environment setup facilitates the correct installation of ATS for NWDAF.

Network Access

The Kubernetes cluster hosts must have network access to the following:

- Local docker image repository, where the OCATS NWDAF images are available.
  To verify if the Kubernetes cluster hosts have network access to the local docker image repository, retrieve any image with tag name to check connectivity by running the following command:

  ```
  docker pull <docker-repo>/<image-name>:<image-tag>
  ```

  Where, `docker-repo` is the IP address or host name of the repository, `image-name` is the docker image name and `image-tag` is the tag the image used for the NWDAF pod.

- Local helm repository, where the OCATS NWDAF helm charts are available.
  To verify if the Kubernetes cluster hosts have network access to the local helm repository, run the following command:

  ```
  helm repo update
  ```

Client Machine Requirement

Listed below are the Client Machine requirements for a successful ATS installation for NWDAF:

- Network access to the Helm repository and Docker image repository.
- Helm repository must be configured on the client.
- Network access to the Kubernetes cluster.
- The environment settings to run the `kubectl` and docker commands. The environment should have privileges to create a namespace in the Kubernetes cluster.
- The Helm client must be installed. The environment should be configured such that the Helm install command deploys the software in the Kubernetes cluster.

cnDBTier Requirement

NWDAF supports cnDBTier in a vCNE environment. cnDBTier must be up and active in case of containerized CNE. For more information, see *Oracle Communications cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

> **Note**
>
> If the environment has cnDBTier 23.2.0 installation, follow the instruction below:
>
> - If cnDBTier 23.2.0 release is installed, set the *ndb_allow_copying_alter_table* parameter to 'ON' in the cnDBTier custom values *dbtier_23.2.0_custom_values_23.2.0.yaml* file and perform cnDBTier upgrade before install, upgrade, or any fault recovery procedure is performed for OCNWDAF. Set the parameter to its default value, 'OFF' once the activity is completed and perform the cnDBTier upgrade to apply the parameter changes.
> - To perform cnDBTier upgrade, see *Oracle Communications Cloud Native Core, cnDBTier Installation, Upgrade, and Fault Recovery Guide*.

Oracle Communications Network Analytics Data Director (OCNADD) Requirements

Oracle Communications Network Analytics Data Director (OCNADD) serves as one of the data sources for the OCNWDAF. If OCNADD is configured as a data source, ensure the following prerequisites are met before OCNWDAF installation:

- OCNADD is setup and running.
- ACL feed is enabled on OCNADD as the required data source.
- Run OCNWDAF *gen_certs* script under */scripts/gen_certs.sh*.

> **Note**
>
> Configure the ACL topic certificate from the OCNADD Kafka Cluster in the OCNWDAF Kafka Cluster to enable secure data flow between OCNADD and OCNWDAF.

For more information, see *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide* and *Oracle Communications Networks Data Analytics Function Installation and Fault Recovery Guide*.

Analytics Database

This database is based on MySQL cluster and stores relational and time-series data. The relational data represents all the objects within the telecommunication network, such as UEs, slices, cells, NFs, and so on and their relationships with each other. The time-series data represents all the KPIs, measurements, and event data collected over time and used in streaming analytics and training ML models.

> **Note**
>
> The deployment of the Mysql Innodb cluster is based on the variable *dbConfigStatus* present in the *values.yaml* file under */helmchart*.

For more information, see *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide* and *Oracle Communications Networks Data Analytics Function Installation and Fault Recovery Guide*.

3.1.3 Resource Requirements

This section describes the ATS resource requirements for NWDAF.

NWDAF Pods Resource Requirements Details

This section describes the resource requirements, which are needed to deploy NWDAF ATS successfully. The following table describes the total resource usage for:

- NWDAF Suite
- NWDAF Notification Consumer Simulator

Table 3-2 NWDAF Pods Resource Requirements Details

Microservice	vCPUs Required per Pod	Memory Required per Pod (GB)	Storage PVC Required per Pod (GB)	Replicas (regular deployment)	Replicas (ATS deployment)	CPUs Required - Total	Memory Required - Total (GB)	Storage PVC Required - Total (GB)
ocn-ats-nwdaf-service	4	8	0	1	1	4	8	0
ocn-ats-nwdaf-notify-service	2	2	0	1	1	2	2	0

3.1.4 Downloading the ATS Package

Locating and Downloading ATS Images

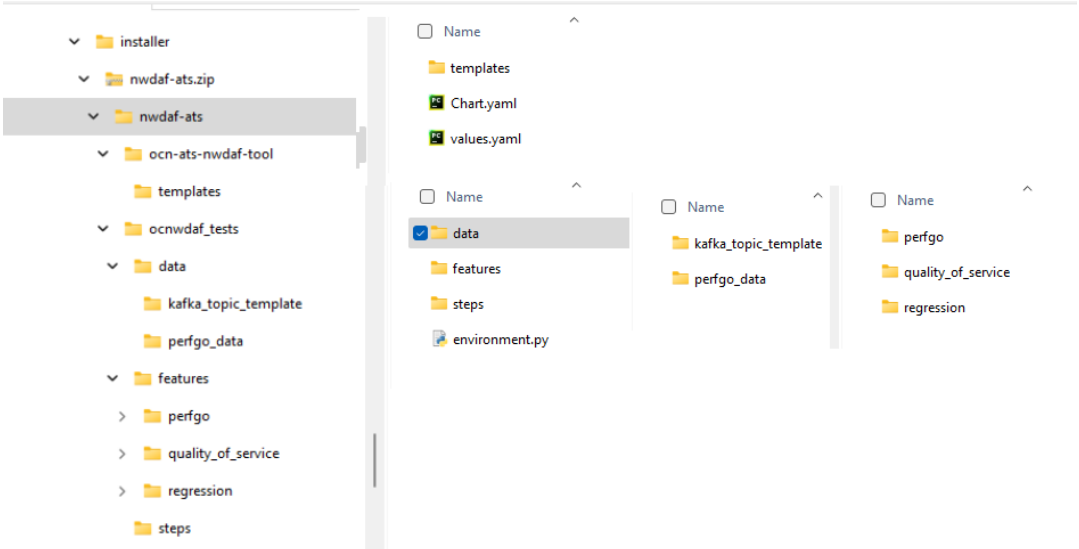
To locate and download the ATS image from MOS:

1. Log in to My Oracle Support using the appropriate credentials.
2. Select the **Patches & Updates** tab.
3. In the **Patch Search** Window, click **Product or Family (Advanced)**.
4. Enter `Oracle Communications Cloud Native Core Network Data Analytics Function` in the **Product** field.
5. From the **Release** drop-down, select "**Oracle Communications Cloud Native Core Network Data Analytics Function <release_number>**" where, `<release_number>` indicates the required release number of OCNWDAF.
6. Click **Search**. The **Patch Advanced Search Results** list appears.
7. Select the required ATS patch from the list. The Patch Details window appears.
8. Click **Download**. The File Download window appears.
9. Click the **<p********_<release_number>_Tekelec>.zip** file to download the OCNWDAF ATS package file.
10. Extract the release package ZIP file. The package is named as *nwdaf-pkg-<marketing-release-number>.tgz*. For example, *nwdaf-pkg-24.1.0.0.tgz*.

11. Untar the OCNWDAF package file to the specific directory, *tar -xvf nwdaf-pkg-<marketing-release-number>.tgz*. The OCNWDAF directory has the following package structure:

```
# Root
- images
  - tar of images
  - sha 256 of images
- troubleshooting/
    - nfDataCapture.sh
- ocn-nwdaf-helmChart/
    - helmChart
        - templates
        - charts
        - values.yaml
        - charts.yaml
        - nwdaf-pre-installer.tar.gz
    - simulator-helmChart
        - templates
        - charts
        - values.yaml
        - charts.yaml
 - nwdaf-ats/
    - ocn-ats-nwdaf-tool
        -templates
    - ocnwdaf_tests
        -data
            - kafka_topic_template
            - perfgo_data
        -features
            - perfgo
            - regression
            - quality of service
        -steps
```

Figure 3-1 OCNWDAF Folder Structure



3.1.5 Pushing the Images to Customer Docker Registry

Follow the procedure described below to push the NWDAF ATS docker images to the docker repository:

Prerequisites

- Oracle Linux 8 environment
- NWDAF 24.1.0.0.0 package

> **Note**
>
> The NWDAF deployment package includes:
>
> - Ready-to-use docker images in the images `tar` or `zip` file.
> - Helm charts to help orchestrate Containers in Kubernetes.

The communication between NWDAF service pods is preconfigured in the Helm charts. The NWDAF ATS uses the following services:

Table 3-3 NWDAF ATS Services

Service Name	Docker Image Name	Image Tag
`ocats-nwdaf`	`ocats-nwdaf`	`24.1.0.0.0`
`ocats-nwdaf-notify`	`ocats-nwdaf-notify`	`24.1.0.0.0`

1. Verify the checksums of tarballs mentioned in file `Readme.txt`.
2. Run the following command to extract the contents of the tar file:

   ```
   tar -xvf nwdaf-pkg-<marketing-release-number>.tgz
   ```

   Or

   To extract the files, run the command:

   ```
   unzip  nwdaf-pkg-<marketing-release-number>.zip
   ```

   The `nwdaf-pkg-<marketing-release-number>.tar` file contains the following NWDAF ATS images:

   - `ocats-nwdaf-notify`
   - `ocats-nwdaf`
3. Navigate to the folder path *./installer*, and extract the zip file. Run the following command:

   ```
   unzip nwdaf-ats.zip
   ```

The zip folder contains the following files:

```
ocn-ats-nwdaf-tool
            ngnix
            templates
        ocnwdaf_tests
            data
            kafka_topic_template
            perfgo_data
            features
                perfgo
                regression
                quality_of_service
            steps
```

4. Run the following command to push the Docker images to the Docker Repository:

   ```
   docker load --input <image_file_name.tar>
   ```

   Example:

   ```
   docker load --input images
   ```

5. Run the following command to push the NWDAF ATS Docker files to the Docker Registry:

   ```
   docker tag <image-name>:<image-tag> <docker-repo>/<image-name>:<image-tag>
   docker push <docker_repo>/<image_name>:<image-tag>
   ```

   Where, `<docker-repo>` indicates the repository where the downloaded images can be pushed.

6. Run the following command to verify if the images are loaded:

   ```
   docker images
   ```

7. Run the following command to push the Helm charts to the Helm repository:

   ```
   helm cm-push --force <chart name>.tgz <helm repo>
   ```

3.1.6 Configuring ATS

This section describes how to configure ATS for NWDAF.

3.1.6.1 Creating and Verifying NWDAF Console Namespaces

This section explains how to create a new namespace or verify an existing namespace in the system.

Run the following command to verify if the required namespace exists in the system:

```
$ kubectl get namespaces
```

If the namespace exists, continue with the NWDAF ATS installation. If the required namespace is not available, create a namespace using the following command:

```
$ kubectl create namespace <required namespace>
```

For example:

```
$ kubectl create namespace ocats-nwdaf
```

Naming Convention for Namespaces:

The naming convention for namespaces must:

- start and end with an alphanumeric character
- contain a maximum of "63" characters
- contain only alphanumeric characters or '-'

3.1.6.2 Updating values.yaml File

Update the values.yaml file before you deploy NWDAF ATS. To update the values.yaml file:

1. In the installation package, navigate to `*root/installer/nwdaf-ats/ocn-ats-nwdaf-tool`
2. Edit the values.yaml file.
   For example:

   `vim values.yaml`

   Update the following parameters in the values.yaml file:

   - `imageRegistry`: Provide the registry where the images are located.
   - `imageVersion`: Verify the value is 24.1.0.0.0.

> **Note**
>
> Ensure that the image registry path is correct, and is pointing to the docker registry where the ATS docker images are located.

3.1.6.3 Deploying NWDAF ATS in the Kubernetes Cluster

To deploy ATS, perform the following steps:

1. The `values.yaml` file is located inside `ocn-ats-nwdaf-tool` directory. The namespace, docker image or tag can be updated in the `values.yaml` file.
2. Run the following command to deploy the NWDAF ATS and its consumers in the same namespace where NWDAF suite is installed:

   ```
   helm install <installation name> <path to the chart directory> -
   n $K8_NAMESPACE
   ```

   For example:

   ```
   helm install ocnwdaf-ats ocn-ats-nwdaf-tool/
   ```

> **Note**
>
> Ensure NWDAF ATS is installed in the same namespace where the NWDAF suite is installed.

3. To perform Helm installation with proxy command, run the following command:

```
helm install <installation name> <path to the chart directory> -
n $K8_NAMESPACE
--set ocatsNwdaf.config.env.JAVA_OPTS="\\-Dhttps\\.proxyHost\
\=<proxy_domain>\\ \\-Dhttps\\.proxyPort\\=<proxy_port>\\ \\-Dhttp\
\.nonProxyHosts\\=localhost\\,127.0.0.1\\,\\<no_proxy_host>"
```

For example:

```
helm install ocnwdaf-ats ocn-ats-nwdaf-tool/ --set
ocatsNwdaf.config.env.JAVA_OPTS="\\-Dhttps\\.proxyHost\\=www-
proxy.us.oracle.com\\ \\-Dhttps\\.proxyPort\\=80\\ \\-Dhttp\\.nonProxyHosts\
\=localhost\\,127.0.0.1\\,\\.oracle.com\\,\\.oraclecorp\\.com"
```

> **Note**
>
> Provide the **ocatsNwdaf.config.env.JAVA_OPTS** field in the proxy configuration. This allows access to the internet to download the required plugins and components required for a successful ATS installation.

4. Run the following command to verify the ATS deployment status:

```
kubectl get deployments -n <namespace_name>
```

3.1.6.4 Verifying ATS Deployment

Run the following command to verify ATS deployment:

```
helm status <release_name>
```

Once ATS is deployed, run the following commands to check the pod and service deployment:

To check pod deployment, run the command:

```
kubectl get pod –n <namespace_name>
```

To check service deployment, run the command:

```
kubectl get service -n <namespace_name>
```

Once the installation (service and pod deployment) is successfully running, track the progress of the ATS Jenkins preconfiguration process, run the following command:

```
kubectl exec -it <ats_pod_name> -- tail -f /var/lib/jenkins/.jenkins/jenkins-configure.log
```

For example:

```
kubectl exec -it ocats-nwdaf-deploy-787d4f5f84-5vmv5 -- tail -f /var/lib/jenkins/.jenkins/jenkins-configure.log
```

Wait for the preconfiguration process to complete, the following message is displayed:

```
Jenkins configuration finish successfully
```

3.2 Installing ATS for OCNADD

The Oracle Communications Network Analytics Data Director (OCNADD) can be installed with or without TLS (Transport Layer Security) Support feature. To enable TLS feature support, see Enabling TLS and Enable TLS Support on OCNADD ATS before proceeding with ATS installation.

Installing ATS for OCNADD

Following are the ATS installation procedures for Oracle Communications Network Analytics Data Director (OCNADD):

1. Resource Requirements
2. Downloading the ATS Package
3. Pushing the Images to Customer Docker Registry
4. Configuring ATS
5. Deploying ATS and Stub in Kubernetes Cluster
6. Verifying ATS Deployment

3.2.1 Resource Requirements

This section describes the ATS resource requirements for OCNADD.

Overview - Total Number of Resources

The following table describes the overall resource usage in terms of CPUs and memory for the following:

- OCNADD SUT
- ATS

Table 3-4 OCNADD - Total Number of Resources

Resource Name	CPU	Memory (GB)
OCNADD SUT Totals	52	308
ATS Totals	10	14

Table 3-4 (Cont.) OCNADD - Total Number of Resources

Resource Name	CPU	Memory (GB)
Grand Total OCNADD ATS	62	322

OCNADD Pods Resource Requirement Details

This section describes the resource requirements, which are needed to deploy OCNADD ATS successfully.

Table 3-5 OCNADD Pods Resource Requirement Details

OCNADD Service	CPUs Required per Pod	Memory Required per Pod (GB)	# Replicas (regular deployment)	# Replicas (ATS deployment)	CPUs Required - Total	Memory Required - Total (GB)
ocnaddconfiguration	1	1	1	1	1	1
ocnaddalarm	1	1	1	1	1	1
ocnaddadmin	1	1	1	1	1	1
ocnaddhealthmonitoring	1	1	1	1	1	1
ocnadduirouter	1	1	1	1	1	1
ocnaddscpaggregation	2	2	1	1	2	2
ocnaddnrfaggregation	2	2	1	1	2	2
ocnaddseppaggregation	2	2	1	1	2	2
ocnaddadapter	3	4	2	1	3	4
ocnaddkafka	6	64	4	4	24	256
zookeeper	1	2	3	3	3	6
ocnaddgui	2	1	1	1	2	1
ocnaddfilter	3	3	1	1	3	3
ocnaddcorrelation	3	24	1	1	3	24
ocnaddredundancyagent	2	3	1	1	3	3
OCNADD SUT Totals					**52 CPU**	**308 GB**

For more information about OCNADD Pods Resource Requirements, see the "Resource Requirements" section in *Oracle Communications Network Analytics Data Director Installation, Upgrade, and Fault Recovery Guide*.

ATS Resource Requirement details for OCNADD

This section describes the ATS resource requirements, which are needed to deploy OCNADD ATS successfully.

Table 3-6 ATS Resource Requirement Details

Microservice	CPUs Required per Pod	Memory Required per Pod (GB)	# Replicas (regular deployment)	# Replicas (ATS deployment)	CPUs Required - Total	Memory Required - Total (GB)
ATS Behave	2	1	1	1	2	1

Table 3-6 (Cont.) ATS Resource Requirement Details

Microservice	CPUs Required per Pod	Memory Required per Pod (GB)	# Replicas (regular deployment)	# Replicas (ATS deployment)	CPUs Required - Total	Memory Required - Total (GB)
OCNADD Producer Stub (SCP,NRF,SEPP)	6	12	1	1	6	12
OCNADD Consumer Stub	2	1	1	1	2	1
ATS Totals					**10**	**14**

3.2.2 Downloading the ATS Package

Locating and Downloading ATS Images

To locate and download the ATS Image from MOS:

1. Log in to My Oracle Support using the appropriate credentials.
2. Select the **Patches & Updates** tab.
3. In the **Patch Search** window, click **Product or Family (Advanced)**.
4. Enter *Oracle Communications Network Analytics Data Director* in the **Product** field.
5. Select *Oracle Communications Network Analytics Data Director <release_number>* from the **Release** drop-down.
6. Click **Search**. The **Patch Advanced Search Results** list appears.
7. Select the required ATS patch from the list. The Patch Details window appears.
8. Click **Download**. The File Download window appears.
9. Click the **<p********_<release_number>_Tekelec>.zip** file to download the OCNADD ATS package file.
10. Untar the zip file to access all the ATS Images. The **<p********_<release_number>_Tekelec>.zip** directory has following files:

```
ocats-ocnadd-tools-pkg-24.1.0.tgz
ocats-ocnadd-tools-pkg-24.1.0-README.txt
ocats-ocnadd-tools-pkg-24.1.0.tgz.sha256
ocats-ocnadd-custom-configtemplates-24.1.0.zip
ocats-ocnadd-custom-configtemplates-24.1.0-README.txt
```

The `ocats-ocnadd-tools-pkg-24.1.0-README.txt` file has all the information required for the package. The `ocats-ocnadd-tools-pkg-24.1.0.tgz` file has the following images and charts packaged as tar files:

```
ocats-ocnadd-tools-pkg-24.1.0.tgz

| |_ _ _ocats-ocnadd-pkg-24.1.0.tgz

| |_ _ _ _ _ _ ocats-ocnadd-24.1.0.tgz (Helm Charts)

| |_ _ _ _ _ _ ocats-ocnadd-image-24.1.0.tar (Docker Images)
```

```
| |_ _ _ _ _ _ OCATS-ocnadd-Readme.txt

| |_ _ _ _ _ _ ocats-ocnadd-24.1.0.tgz.sha256

| |_ _ _ _ _ _ ocats-ocnadd-image-24.1.0.tar.sha256

| |_ _ _ _ _ _ ocats-ocnadd-data-24.1.0.tgz (ATS test scripts and Jenkins
data)

| |_ _ _ _ _ _ ocats-ocnadd-data-24.1.0.tgz.sha256

| | |_ _ _ocstub-ocnadd-pkg-24.1.0.tgz

|_ _ _ _ _ _ ocstub-ocnadd-24.1.0.tgz (Helm Charts)

|_ _ _ _ _ _ ocstub-ocnadd-image-24.1.0.tar (Docker Images)

|_ _ _ _ _ _ OCSTUB-ocnadd-Readme.txt

|_ _ _ _ _ _ ocstub-ocnadd-24.1.0.tgz.sha256

|_ _ _ _ _ _ ocstub-ocnadd-image-24.1.0.tar.sha256
```

In addition to the above images and charts, there is an `ocats-ocnadd-custom-configtemplates-24.1.0.zip` file in the package file.

```
ocats-ocnadd-custom-configtemplates-24.1.0.zip

      |

      |_ _ _ocats-ocnadd-custom-values_24.1.0.yaml (Custom values file for
installation)

      |

      |_ _ _ocats_ocnadd_custom_serviceaccount_24.1.0.yaml (Template to
create custom service account)
```

11. Copy the tar file to the CNE, OCI, or Kubernetes cluster where you want to deploy ATS.

3.2.3 Pushing the Images to Customer Docker Registry

Preparing to deploy ATS and Stub Pod in Kubernetes Cluster

To deploy ATS and Stub Pod in Kubernetes Cluster:

1. Run the following command to extract tar file content.

```
tar -xvf ocats-ocnadd-tools-pkg-24.1.0.tgz
```

The output of this command is:

```
ocats-ocnadd-pkg-24.1.0.tgz
ocstub-ocnadd-pkg-24.1.0.tgz
ocats-ocnadd-custom-configtemplates-24.1.0.zip
```

2. Run the following command to extract the helm charts and docker images of ATS.

```
tar -xvf ocats-ocnadd-pkg-24.1.0.tgz
```

The output of this command is:

```
ocats-ocnadd-24.1.0.tgz

ocats-ocnadd-24.1.0.tgz.sha256
ocats-ocnadd-data-24.1.0.tgz
ocats-ocnadd-data-24.1.0.tgz.sha256
ocats-ocnadd-image-24.1.0.tar
ocats-ocnadd-image-24.1.0.tar.sha256
OCATS-ocnadd-Readme.txt
```

> **Note**
>
> The `ocats-ocnadd-Readme.txt` file has all the information required for the package.

3. Run the following command to untar the ocstub package.

```
tar -xvf ocstub-ocnadd-pkg-24.1.0.tgz
```

The output of this command is:

```
ocstub-ocnadd-image-24.1.0.tar
ocstub-ocnadd-24.1.0.tgz.sha256
ocstub-ocnadd-image-24.1.0.tar.sha256
ocstub-ocnadd-24.1.0.tgz
OCSTUB-ocnadd-Readme.txt
OCSTUB_OCNADD_Installation_Readme.txt
```

4. Run the following command to extract the content of the custom configuration templates:

```
unzip ocats-ocnadd-custom-configtemplates-24.1.0.zip
```

The output of this command is:

```
ocats-ocnadd-custom-values_24.1.0.yaml (Custom yaml file for deployment of
OCATS-OCNADD)
ocats_ocnadd_custom_serviceaccount_24.1.0.yaml (Custom yaml file for
service account creation to help the customer if required)
```

5. Run the following commands in your cluster to load the ATS docker image, 'ocats-ocnadd-image-24.1.0.tar:

```
$ podman load -i ocats-ocnadd-image-24.1.0.tar
```

6. Run the following commands to tag the imported image and push it to the local registry:

```
$ podman tag docker.io/<image-tag> <local_registry>/<image-name>:<image-tag>
$ podman push <local_registry>/<image-name>:<image-tag>
```

7. Run the following commands in your cluster to load the Stub docker images ocstub-ocnadd-image-24.1.0.tar:

```
$ podman load -i ocstub-ocnadd-image-24.1.0.tar
```

8. Run the following commands to tag each imported image and push it to the local registry:

```
$ podman tag docker.io/<image-tag> <local_registry>/<image-name>:<image-tag>
$ podman push <local_registry>/<image-name>:<image-tag>
```

9. Update the image name and tag in the ocats-ocnadd-custom-values.yaml and ocnaddsimulator/values.yaml files of simulator Helm as required. For ocats-ocnadd-custom-values.yaml update the 'image.repository' with respective local_registry. For ocnaddsimulator/values.yaml update the 'repo.REPO_HOST_PORT' and 'initContainers.repo.REPO_HOST_PORT' with respective local_registry.

3.2.4 Configuring ATS

3.2.4.1 Enabling Static Port

1. To enable static port:
   - In the **ocats-ocnadd-custom-values.yaml** file under service section, set the **staticNodePortEnabled** parameter value to 'true' and **staticNodePort** parameter value with valid nodePort.

```
service:
  customExtension:
    labels: {}
    annotations: {}
  type: LoadBalancer
  port: "8080"
  staticNodePortEnabled: true
  staticNodePort: "32385"
  staticLoadBalancerIPEnabled:false
  staticLoadBalancerIP: ""
```

> **Note**
>
> ATS supports static port. By default, this feature is not enabled.

> **Note**
>
> To enable `staticLoadBalancerIP`, set the `staticLoadBalancerIPEnabled` parameter value to 'true' and `staticLoadBalancerIP` parameter value with valid `LoadBalancer IP` value. By default, this is set to false.

3.2.4.2 Enabling TLS

To install ATS with TLS support, update the following parameters to "`true`" in the *ocats-ocnadd-custom-values.yaml* file:

- `atsGuiTLSEnabled`
- `atsCommunicationTLSEnabled`

For example:

```
atsGuiTLSEnabled: false               ## --> updated it with 'true'
atsCommunicationTLSEnabled: false     ## --> updated it with 'true'
```

3.2.5 Deploying ATS and Stub in Kubernetes Cluster

> **Note**
>
> It is important to ensure that the ATS and Stub are deployed as follows:
>
> - In **Centralized Deployment** mode (fresh deployments), the ATS and Stub must be deployed in the **Worker Group** namespace.
> - In **Non-Centralized Deployment** mode (when the OCNADD is upgraded from any older release), the ATS and Stub should be deployed in the **default** OCNADD namespace.
> - When OCNADD is deployed as a two-site georedundant deployment, ATS and Stub should be deployed on both Primary and Secondary sites.
> - OCNADD-ATS suit does not support mTLS, ensure that mTLS is set to "false" during OCNADD deployment.

For the test cases to run successfully, ensure the *intraTLSEnalbled* parameter value in the Jenkins pipeline script is identical to the value in the OCNADD deployment. The Alert Manager does not support HTTPS connections. When *intraTLSEnalbled* is set to *true*, the following Alert Manager test cases may fail:

- Verify the OCNADD_CONSUMER_ADAPTER_SVC_DOWN alert, this alert is raised when deployments of OCNADD are broken.
- Verify the OCNADD_ADMIN_SVC_DOWN alert, this alert is raised when deployments of OCNADD are broken.
- Verify the OCNADD_NRF_AGGREGATION_SVC_DOWN alert, this alert is raised when deployments of OCNADD are broken.
- Verify the OCNADD_SCP_AGGREGATION_SVC_DOWN alert, this alert is raised when deployments of OCNADD are broken.

- Verify the OCNADD_ALARM_SVC_DOWN alert, this alert is raised when deployments of OCNADD are broken.
- Verify the OCNADD_HEALTH_MONITORING_SVC_DOWN alert, this alert is raised when deployments of OCNADD are broken.

ATS and Stub Support Helm Deployment

Choose the namespace to deploy the ATS and Stub based on the OCNADD deployment mode, see Note.

If the namespace does not exist, run the following command to create a namespace:

```
kubectl create namespace <namespace_name>
```

> **Note**
>
> - It is recommended to use the `<release_name>` as `ocnadd-sim` while installing stubs.
> - The ATS deployment with OCNADD does not support the Persistent Volume (PV) feature. Therefore, the default value of the `deployment.PVEnabled` parameter in `ocats-ocnadd-custom-values.yaml` must not be changed. By default, the parameter value is set to **false**.

Deploying ATS:

Run the following command:

```
helm install -name <release_name> ocats-ocnadd-23.4.0.tgz --namespace
<namespace_name> -f <values-yaml-file>
```

Example:

```
helm install -name ocats ocats-ocnadd-23.4.0.tgz --namespace ocnadd-deploy -f
ocats-ocnadd-custom-values.yaml
```

Deploying Stubs:

> **Note**
>
> Before you deploy stubs, update the parameter *oraclenfproducer.REST_BASED_TRAFFIC* to *true* in the *ocnaddsimulator/values.yaml* file. The default value of the parameter is *false*.

```
helm install -name <release_name> <ocstub-ocnadd-chart> --namespace
<namespace_name>
```

> **Note**
>
> For more details about installing the stub, refer to the `OCSTUB_OCNADD_Installation_Readme.txt` file.

Example:

```
helm install -name ocnadd-sim ocnaddsimulator --namespace ocnadd-deploy
```

3.2.6 Verifying ATS Deployment

Run the following command to verify ATS deployment.

```
helm status <release_name> -n <namespace>
```

Once ATS and Stub are deployed, run the following commands to check the pod and service deployment:

To check pod deployment:

```
kubectl get pod -n ocnadd-deploy
```

To check service deployment:

```
kubectl get service -n ocnadd-deploy
```

3.2.7 Enable TLS Support on OCNADD ATS

To enable TLS on OCNADD ATS follow the steps listed below:

- Generate *.jks* File for the Jenkins Server
- Enable the ATS GUI with HTTPS during Installation
- Installing ATS for OCNADD

Generate *.jks* File for the Jenkins Server

For Jenkins to support GUI access through HTTPS, a *.jks* file has to be created. Follow the steps below to create the *.jks* file:

1. **Generate the Root Certificate (*caroot.cer*)**

   If you have CA signed root certificate and key or your own root certificates, use the same certificates.

   If the root certificate (*caroot.cer*) is not there, follow the steps below to create the certificate to create self-signed certificates. The root certificate is added to the Trust Store (of the browser). However, this step is not required for CA signed certificates.

   a. Follow any browser's documentation, for example:
      - **Mozilla Firefox**
        - Navigate to the **Settings** and search for **Certificate**.
        - Click **View Certificates** in the search results.

- The **Certificate Manager** window appears on the screen.
- Navigate to the **Authorities** section and click **Import**. Upload the *caroot* certificate.
- A new window appears on the screen, click **Trust Options** and then click **OK**.
- Save the changes and restart the browser.

- **Google Chrome**
  - Navigate to the **Settings** and search for **Security**.
  - Click **Security** in the search results.
  - Locate **Manage Device Certificates** and click to select the option.
  - The **Certificates** window appears on the screen. Click **Trusted Root Certification Authorities** bar.
  - Import the *caroot* certificate.
  - Save the changes and restart the browser.

b. Use the Root Certificate to sign the Application or ATS certificate.

c. To generate the key *caroot.key*, run the command `openssl genrsa 2048 > caroot.key`.

d. Generate the *caroot* certificate, for example:

```
[cloud-user@star23-bastion-1 edo]$ openssl req -new -x509 -nodes -days
1000 -key caroot.key > caroot.cer
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:IN
State or Province Name (full name) []:KA
Locality Name (eg, city) [Default City]:BLR
Organization Name (eg, company) [Default Company Ltd]:ORACLE
Organizational Unit Name (eg, section) []:CGBU
Common Name (eg, your name or your server's hostname) []:ocats
Email Address []:
[cloud-user@star23-bastion-1 edo]$
```

2. **Generate Application or Client Certificate**

   Follow the steps below to generate an application or client certificate:

   a. Create and edit the *ssl.conf* file as below:
   In the `[alt_names]` section, list the IPs through which the ATS GUI is opened. User can add multiple IPs such as IP.1, IP.2, and so on.

```
[ req ]
default_bits       = 4096
distinguished_name = req_distinguished_name
req_extensions     = req_ext
```

```
[ req_distinguished_name ]
countryName                 = Country Name (2 letter code)
countryName_default         = IN
stateOrProvinceName         = State or Province Name (full name)
stateOrProvinceName_default = KN
localityName                = Locality Name (eg, city)
localityName_default        = BLR
organizationName            = Organization Name (eg, company)
organizationName_default    = ORACLE
commonName                  = Common Name (e.g. server FQDN or YOUR
name)
commonName_max              = 64
commonName_default          = ocats.scpsvc.svc.cluster.local

[ req_ext ]
keyUsage = critical, digitalSignature, keyEncipherment
extendedKeyUsage = serverAuth, clientAuth
basicConstraints = critical, CA:FALSE
subjectAltName = critical, @alt_names

[alt_names]
IP.1 = 127.0.0.1
IP.2 = 10.75.217.5
IP.3 = 10.75.217.76
DNS.1 = localhost
DNS.2 = ocats.scpsvc.svc.cluster.local
```

To access the GUI with DNS ensure that the `commonName_default` value is the same as the configured DNS name and the format is `<service_name>.<namespace>.svc.cluster.local`. User can add multiple DNS'es like DNS.1,DNS.2, and so on.

To support ATS API's, add `127.0.0.1` as `IP.1`.

b. Create a Certificate Signing Request or CSR file, as follows:

```
[cloud-user@star23-bastion-1 ocats]$ openssl req -config ssl.conf -
newkey rsa:2048 -days 1000 -nodes -keyout rsa_private_key_pkcs1.key >
ssl_rsa_certificate.csr
Ignoring -days; not generating a certificate
Generating a RSA private key
...+++++
........+++++
writing new private key to 'rsa_private_key_pkcs1.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [IN]:
State or Province Name (full name) [KA]:
Locality Name (eg, city) [BLR]:
```

```
Organization Name (eg, company) [ORACLE]:
Common Name (e.g. server FQDN or YOUR name) [ocats]:
[cloud-user@star23-bastion-1 ocats]$
```

c. To view all the components of the file, run the command `openssl req -text -noout -verify -in ssl_rsa_certificate.csr`. Verify all the configurations in the file.

d. Sign the .csr file with the root certificate, as below:

```
[cloud-user@star23-bastion-1 ocats]$ openssl x509 -extfile ssl.conf -
extensions req_ext -req -in ssl_rsa_certificate.csr -days 1000 -CA ../
caroot.cer -CAkey ../caroot.key -set_serial 04 > ssl_rsa_certificate.crt
Signature ok
subject=C = IN, ST = KA, L = BLR, O = ORACLE, CN = ocats
Getting CA Private Key
[cloud-user@star23-bastion-1 ocats]$
```

e. Verify if the certificate is correctly signed by root certificate, as below:

```
[cloud-user@star23-bastion-1 ocats]$ openssl verify -CAfile caroot.cer
ssl_rsa_certificate.crt
ssl_rsa_certificate.crt: OK
```

3. Save the application certificate and the root certificates for future use.
4. Add the *caroot.cer* certificate as a trusted author in the browser.
5. The generated application or client certificates cannot be provided directly to the Jenkins server. Follow the steps below:
   - Generate the *.p12 keystore* file, run the following command:

     ```
     [cloud-user@star23-bastion-1 ocats]$ openssl pkcs12 -inkey
     rsa_private_key_pkcs1.key -in ssl_rsa_certificate.crt -export -out
     certificate.p12
     Enter Export Password:
     Verifying - Enter Export Password:
     ```

     You will receive a password prompt, note the password provided for future use.

     – Convert *.p12* file to *.jks* file, run the following command:

       ```
       [cloud-user@star23-bastion-1 ocats]$ keytool -importkeystore -
       srckeystore ./certificate.p12 -srcstoretype pkcs12 -destkeystore
       jenkinsserver.jks -deststoretype JKS
       Importing keystore ./certificate.p12 to jenkinsserver.jks...
       Enter destination keystore password:
       Re-enter new password:
       Enter source keystore password:
       Entry for alias 1 successfully imported.
       Import command completed:  1 entries successfully imported, 0
       entries failed or cancelled
       ```

       You will receive a password prompt, provide the same password used for creating *.p12* file.

- Ensure that both *p12* and *jks* files have the same password.
- Provide the generate *jenkinserver.jks* file to the Jenkins Server.

Enable the ATS GUI with HTTPS during Installation

Follow the procedure below to enable the ATS GUI with HTTPS during installation:

1. To create the *.jks* file, see Generate *.jks* File for the Jenkins Server. You can choose to use the either a CA signed certificate or a self signed certificate.
2. Create a Kubernetes secret by using the files created in the Generate *.jks* File for the Jenkins Server procedure, run the following command:

   ```
   kubectl create secret generic ocats-tls-secret --from-
   file=jenkinsserver.jks --from-file=ssl_rsa_certificate.crt --from-
   file=rsa_private_key_pkcs1.key --from-file=caroot.cer -n scpsvc
   ```

   View the K8s secret, for example:

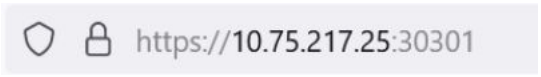
   ```
   [cloud-user@star23-bastion-1 ~]$ kubectl describe secret ocats-tls-secret -
   n scpsvc
   Name:         ocats-tls-secret
   Namespace:    scpsvc
   Labels:       <none>
   Annotations:  <none>

   Type:  Opaque

   Data
   ====
   caroot.cer:                      1147 bytes
   ssl_rsa_certificate.crt:         1424 bytes
   jenkinsserver.jks:               2357 bytes
   rsa_private_key_pkcs1.key:       1675 bytes
   ```

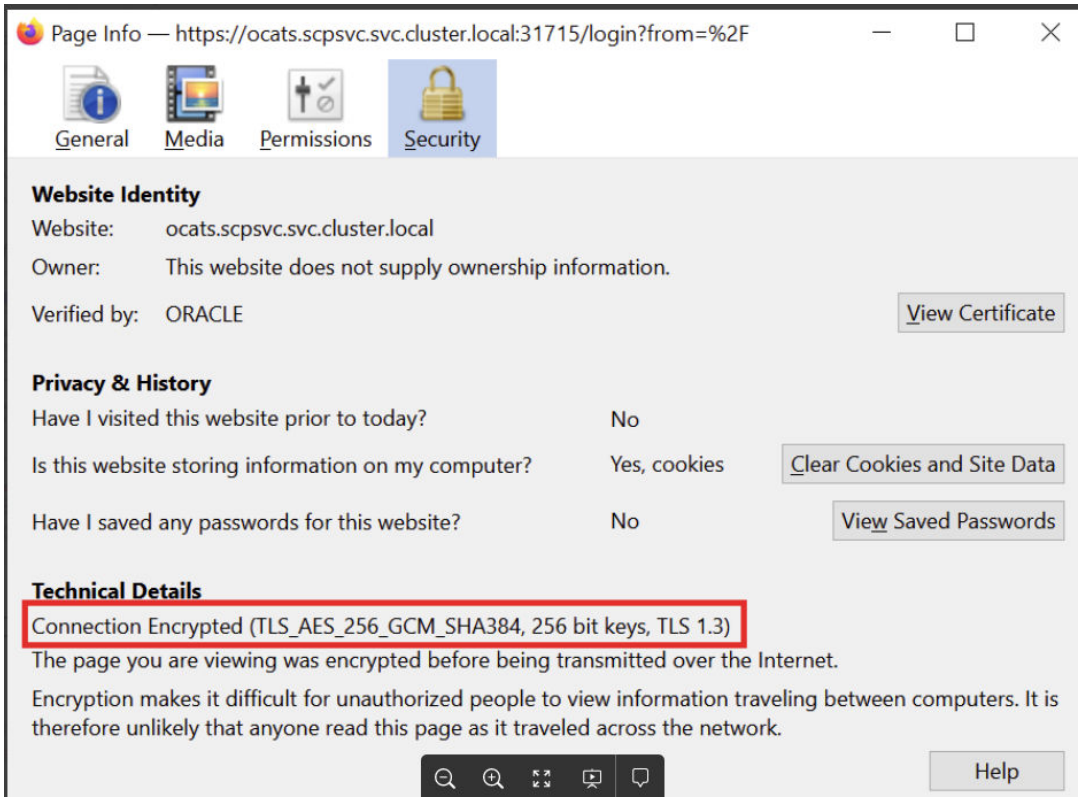
3. After creating the secret, follow the Installing ATS for OCNADD procedure to install the ATS. Once the installation is successfully completed, proceed to the next step.
4. The ATS GUI opens with HTTPS protocol. The link to open the GUI appears as *https://<IP>:<port>*.
   For example:

   Figure 3-2 HTTPS Address

5. In Mozilla Firefox browser, click the **Lock** symbol in the address bar, the page information is displayed as below:

Figure 3-3 Page Information



4
Running Test Cases Using ATS

This section describes how to run test cases using ATS. It includes:

- Running NWDAF Test Cases using ATS
- Running OCNADD Test Cases using ATS

4.1 Running NWDAF Test Cases using ATS

This section describes how to run Networks Data Analytics Function (NWDAF) test cases using ATS.

Prerequisites

To run NWDAF test cases, ensure that the following prerequisites are met:

- The ATS version must be compatible with the NWDAF release.
- The NWDAF and ATS must be deployed in the same namespace.
- The NWDAF must be deployed using the appropriate `values.yaml` file as per the configuration to be tested.
- Ensure a nodePort or LoadBalancer port is available for "ocats-nwdaf-deploy".

Logging into ATS

Running ATS

Before logging in to ATS, you need to ensure that ATS is deployed successfully using HELM charts.

For more information on verifying ATS deployment, see Verifying ATS Deployment .

> **Note**
>
> To modify default log in password, refer to Modifying Login Password.

Build the Jenkins host IP and load the Jenkins tool, run the following command to obtain the ocats-nwdaf-deploy service port:

```
kubectl get svc -n <namespace> | grep ocats-nwdaf-deploy
```

To obtain ocn-ats-nwdaf-tool pod IP:

- Obtain the node and pod name, run the following command:

  ```
  kubectl get pod -o=custom-columns=NODE:.spec.nodeName,NAME:.metadata.name -
  n <namespace> | grep ocats-nwdaf-deploy
  ```

- If an external IP is used, obtain the external Kubernetes node IP. Run the following command:

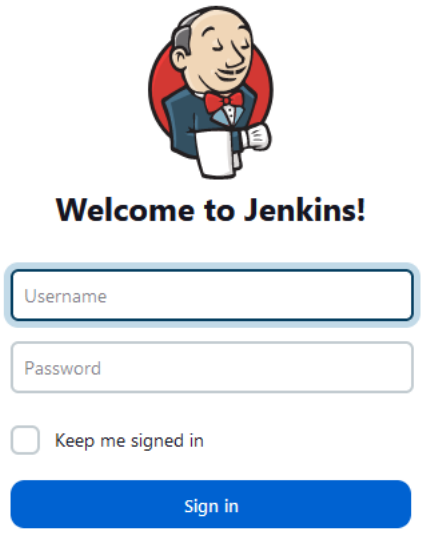
  ```
  kubectl get no -o wide
  ```

- Build the IP host as <External Node IP> : <Host IP>

ATS Login

1. To log in to ATS, open a browser and provide the IP Address and port details as <Worker-Node-IP>:<Node-Port-of-ATS>.

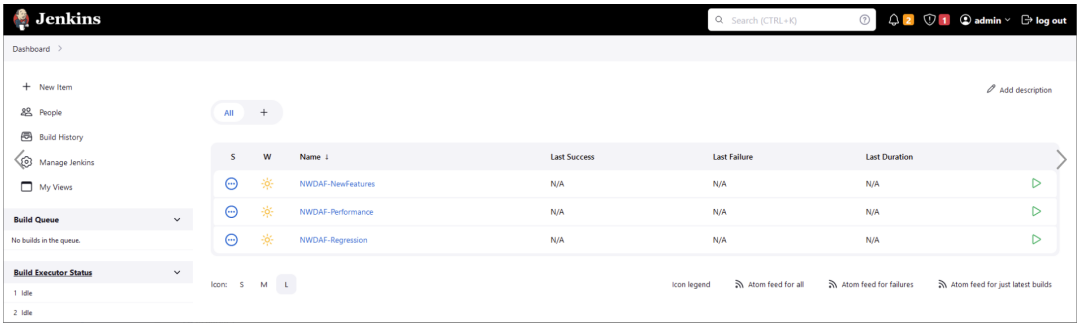
Figure 4-1 ATS Login



To run ATS, enter the login credentials. Click **Sign in**.

2. **Cuztomize Jenkins** page appears. Close this page.
3. **Jenkins is ready!** page appears, click **Start Using Jenkins**.
4. Dashboard screen displaying the NWDAF preconfigured pipelines appears.

Figure 4-2 Pipelines



The NWDAF ATS has three configured pipelines:

- NWDAF-NewFeatures
- NWDAF-Performance
- NWDAF-Regression

Configure NWDAF Pipelines

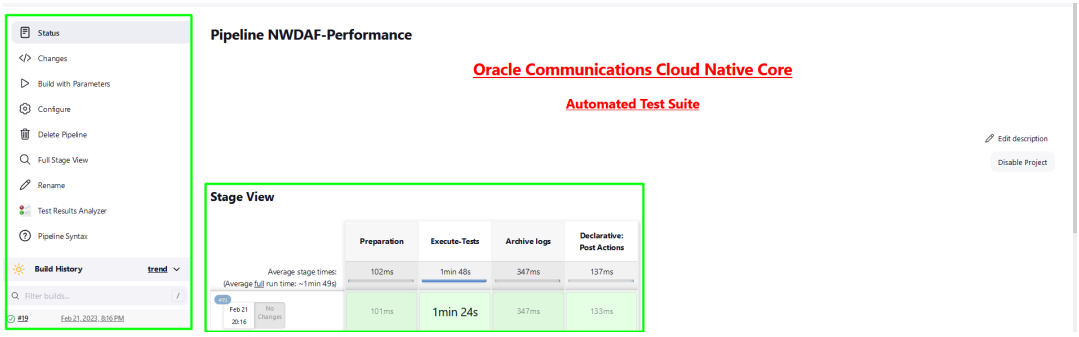
To run configure a pipeline perform the following steps:

Prerequisites

- The user database access includes permission to retrieve, update, and delete information.
- The user has logged in to the Jenkins page with admin credentials.

1. Select the pipeline you want to run the test cases on, click **NWDAF-NewFeatures** in the **Name** column. The following screen appears:

Figure 4-3 Pipeline Configuration



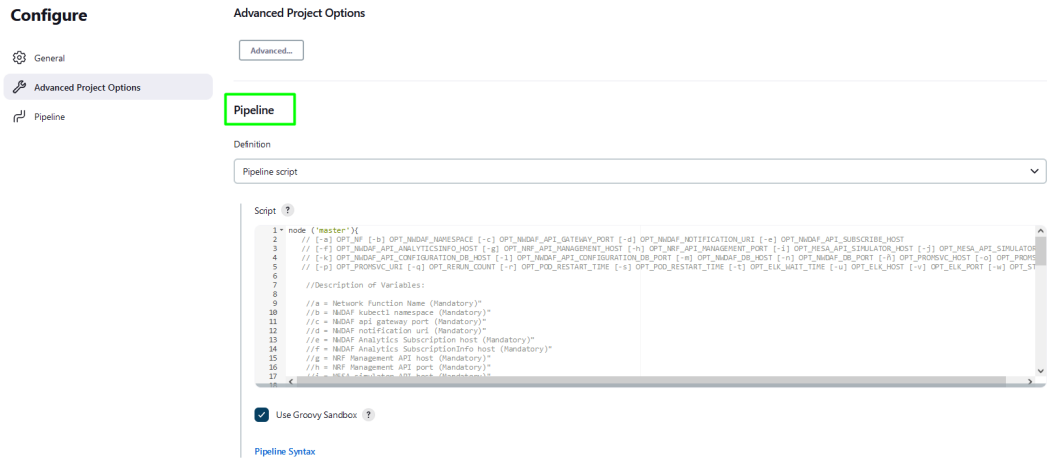
In the above screen:

- Click **Configure** to configure the NWDAF pipeline.
- Click **Build History** box to view all the previous pipeline executions, and the Console Output of each execution.
- The **Full Stage View** represents the previously run pipelines for reference.
- The **Test Results Analyzer** is the plugin integrated into the OCNWDAF-ATS. This option can be used to display the build-wise history of all the executions. It will provide a graphical representation of the past execution together.

2. Click **Configure** to configure the environment parameters of the pipeline. User must wait for the page to load completely. Once the page loads completely, move to the **Pipeline** section:

> **Note**
>
> Make sure that the following page loads completely before you perform any action on it. Also, do not modify any configuration other than shown below.

Figure 4-4 Configure Pipeline



> **Note**
>
> Remove the existing default content of the pipeline script.

3. Depending on the pipeline you want to configure, use the respective script. The following scripts are available at: `/var/lib/jenkins/ocnwdaf_tests/pipeline_scripts`:
   - NewFeatures_Pipeline_script.txt
   - Regression_Pipeline_script.txt
   - Performance_Pipeline_script.txt
4. Update the script of the pipeline you want to configure, and update the parameters according to the ATS and NWDAF environment you want to test.
   a. **- b** : Use this option to update the namespace according to the current namespace being used.
   b. **- c** : Use this option to update the service name of the Ingress gateway according to the current service name of the Ingress gateway being used.
   Run the following command in the NWDAF console to verify the current service name of the Ingress gateway:

   ```
   kubectl get svc -n <namespace_name> | grep ingress
   ```

   For example:

   ```
   kubectl get svc -n mdc-nwdaf-qa | grep ingress
   ```

   Figure 4-5 Ingress Gateway

   ```
   $ kubectl get svc -n mdc-nwdaf-qa | grep ingress
   nwdaf-ingress-gateway                NodePort    10.233.12.1     <none>        80:30277/TCP,81:31960/TCP
   ```

   c. Verify if the same name is assigned to "**-c = NWDAF api gateway**" host field, if not update to match the current name. Use the option "**-c ingress-gateway-service **" .

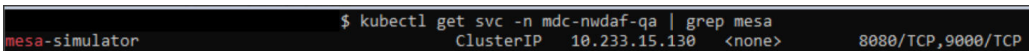
d. **-g** : Use this option to update the service name of the MESA simulator according to the current service name of the MESA simulator being used.
Run the following command in the NWDAF console to verify the current service name of the Mesa simulator:

```
kubectl get svc -n <namespace_name> | grep mesa
```

For example:

```
kubectl get svc -n mdc-nwdaf-qa | grep mesa
```

Figure 4-6 Mesa Simulator



e. Verify if the same name is assigned to "**-g = MESA simulator API**" host field, if not update to match the current name. Use the option "**-g mesa-simulator-service **" .

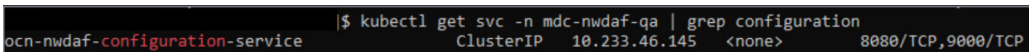
f. **-i** : Use this option to update the service name of the NWDAF Configuration API Host Service according to the current service name of the NWDAF Configuration API Host Service being used.
Run the following command in the NWDAF console to verify the current service name of the NWDAF Configuration API Host Service:

```
kubectl get svc -n <namespace_name> | grep configuration
```

For example:

```
kubectl get svc -n mdc-nwdaf-qa | grep configuration
```

Figure 4-7 Configuration API Host Service



g. Verify if the same name is assigned to "**-i = NWDAF CONFIGURATION API**" host field, if not update to match the current name. Use the option "**-i ocn-nwdaf-configuration-service **" .

5. Update the following parameters according to the database setup:

- **-q** : Use this option to update the NWDAF database host name, it can be FQDN or IP address.
For example:

  **-q 10.75.245.174 **

- **-r** : Use this option to update the NWDAF database port, it can be Cluster IP or Node Port.
For example:

  **-r 32648 **

- **-F** : Use this option to update the InnoDB database host name, it can be FQDN or the IP address.
For example:

**-F 10.75.245.189 **

- **-G** : Use this option to update the InnoDB database port, it can be Cluster IP or Node Port.
  For example:

  **-G 3306 **

6. Update the *-s db_user_id*, *-t db_user_psw*, *inno_db_user_id,* and *inno_db_user_psw*variables with the current data base user credentials, follow the steps listed below:

   a. Provide the userid and password of the database to be encrypted. Navigate to ocats-nwdaf-deploy pod and run the command:

   ```
   kubectl exec -it <pod_name> -n <namespace_name> bash
   ```

   For example:

   ```
   kubectl exec -it ocats-nwdaf-deploy-54f98c447c-l6vxq -n ocnwdaf-ns bash
   ```

   b. From the console, run the following command:

   ```
   cd /env/lib/python3.9/site-packages/ocnnwdaf_lib/db_mgt
   ```

   c. To update the user id run the following command from the console:

   ```
   python -c "from db_connection_mgt import *;
   print(encode_values_data('<user>'))"
   ```

   d. To update the password run the following command from the console:

   ```
   python -c "from db_connection_mgt import *;
   print(encode_values_data('<password>'))"
   ```

   e. Copy and paste the encrypted user id and password into the pipeline script located in the bottom of the page.
   For example:

Figure 4-8 Sample Output

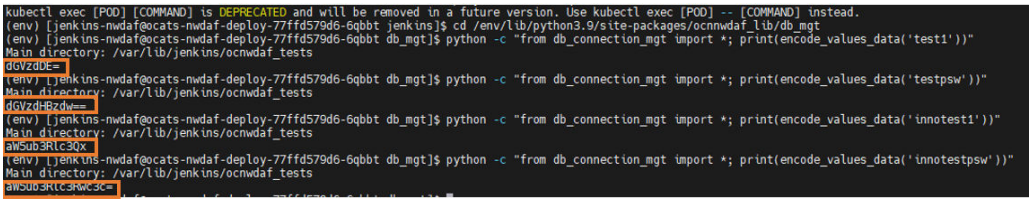


Figure 4-9 Pipeline Script

```
sh /var/lib/jenkins/ocnwdaf_tests/preTestConfig-NewFeatures-NWDAF.sh \
-a NWDAF \
-b ocnwdaf-ns \
-c nwdaf-ingress-gateway \
-d 80 \
-e http://ocats-nwdaf-notify:5000 \
-f http://ocats-nwdaf-notify:5000 \
-g mesa-simulator \
-h 8080 \
-i cap4c-configuration-manager-service \
-j 8080 \
-k 0 \
-l 0 \
-m nwdaf-cap4c-kafka-ui-nodeport-svc \
-n 8080 \
-o 0 \
-p 0 \
-q mysql \
-r 3306 \
-s dGVzdDE= \
-t dGVzdHBzdw== \
-D aW5ub3R1c3Qx \
-E aW5ub3R1c3Rwc3c= \
-F mysql \
-G 3306 \
-u occne-prometheus-server.occne-infra \
-v 80 \
-w /prometheus/api/v1 \
-x 0 \
-y 45 \
-z 0 \
-A occne-elastic-elasticsearch-master.occne-infra \
-B 9200 \
-C yes \
```

Save the values for later use.

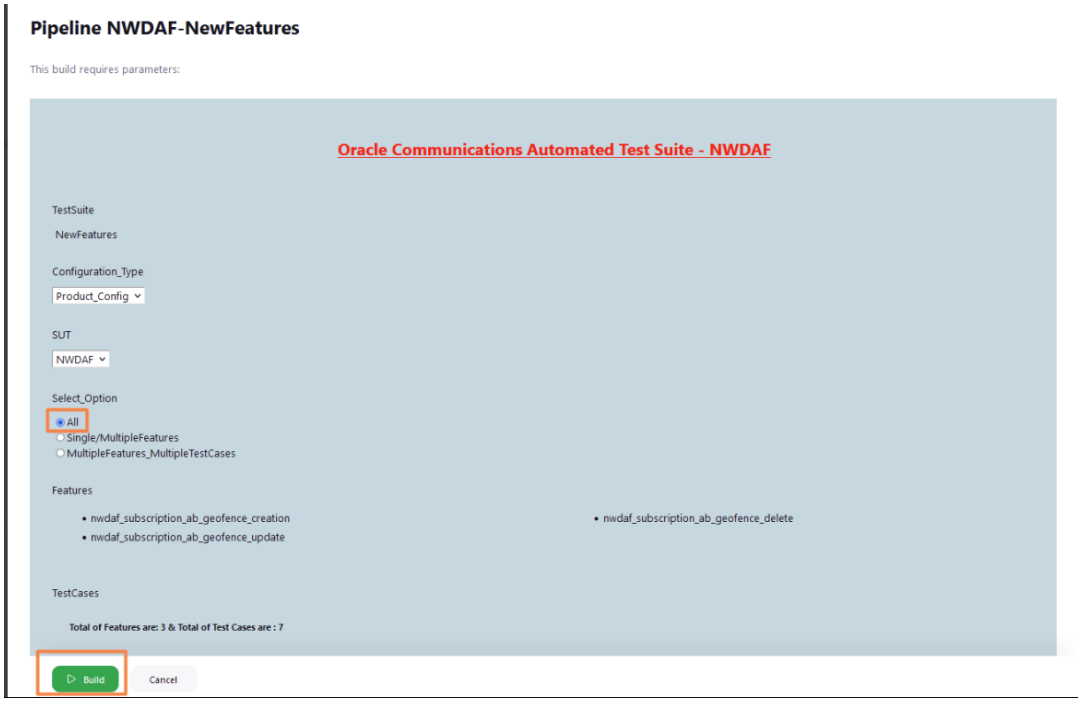
f. Select the **Use Groovy Sandbox** checkbox.

7. Click **Save**. Perform the above steps for **NWDAF-Regression** and **NWDAF-Performance** pipelines.

Run the NWDAF ATS test scrips

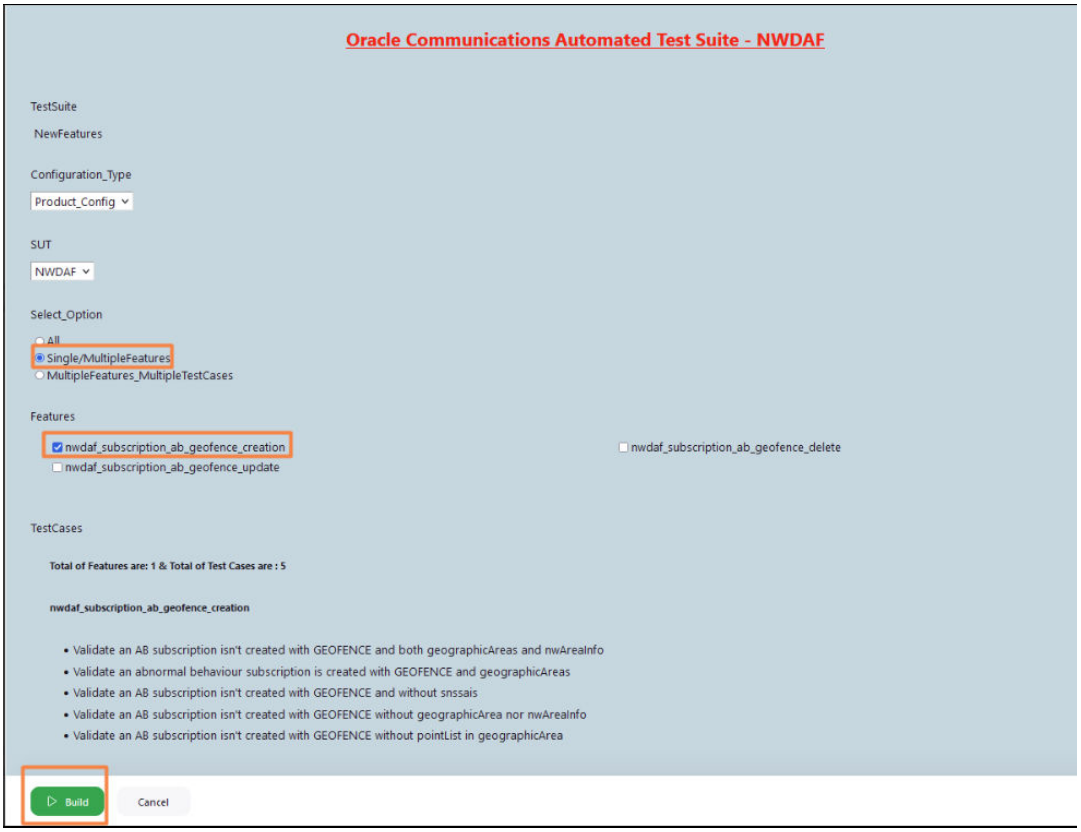
1. To run the test cases, click **Build with Parameters**.
2. The TestSuite multi-option page is displayed, ensure the SUT value is NWDAF.

Figure 4-10 TestSuite Page



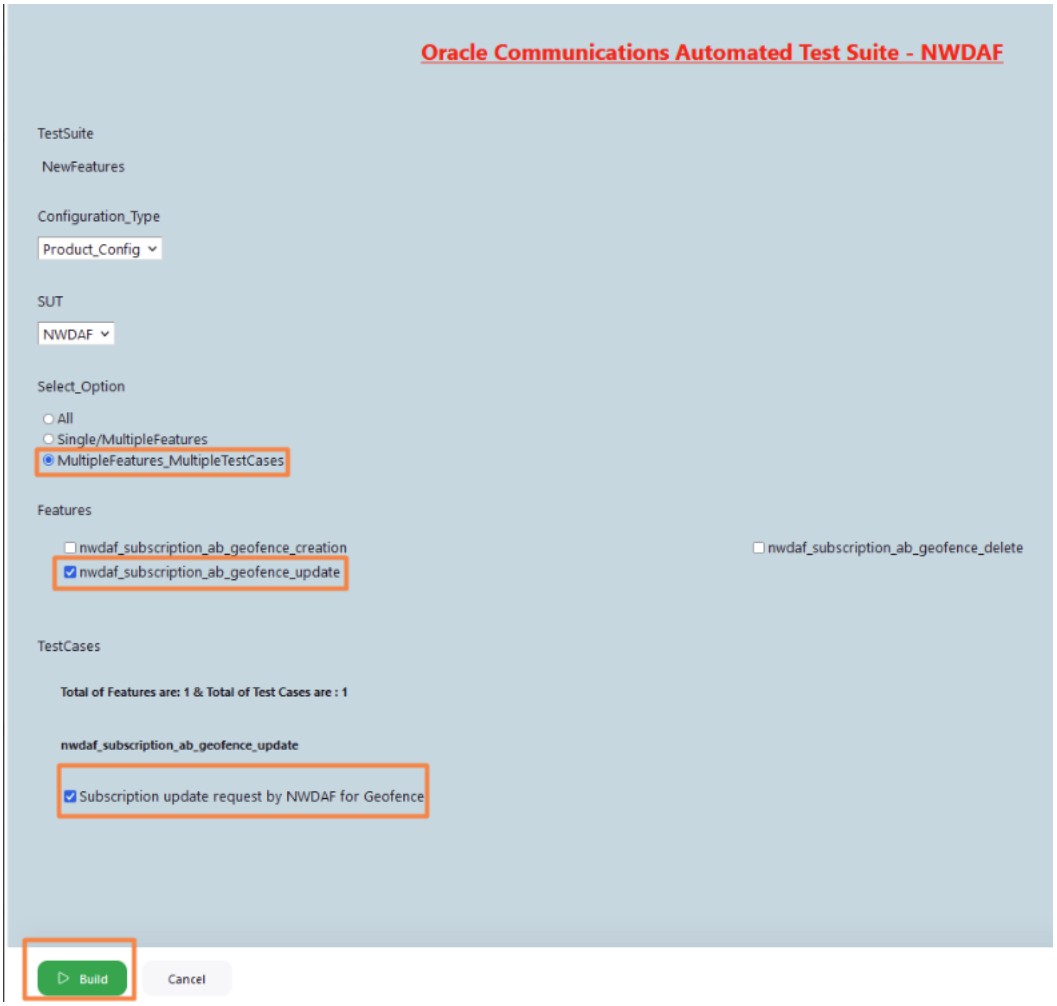
3. To run all scripts available, select **All** and click **Build**.
4. To run scripts on a specific feature, select **Single/MultipleFeatures** option and select the **Feature** you want to run the test on and click **Build**.

Figure 4-11 Single/Multiple Features



5. To run scripts on a specific test scenario, select **MultipleFeatures_MultipleTestCases** option and select the **Feature** and the **TestCases** you want to run the test on and click **Build**.

Figure 4-12 Multiple Features and Testcases

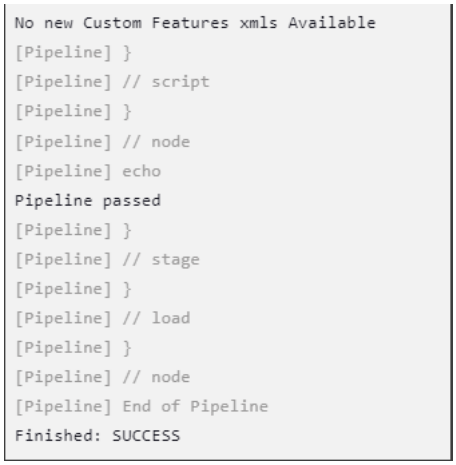


6. The **Build History** menu displays the list of running jobs, select the latest job displayed in the list.
7. Click the **Console Output** option for this job. The jobs progress can be visualized in the log. The job progress also displays results for test cases and pipelines completed.
For example:

Figure 4-13 Sample Output

```
1 feature passed, 0 failed, 0 skipped
3 scenarios passed, 0 failed, 18 skipped
30 steps passed, 0 failed, 162 skipped, 0 undefined
Took 0m45.569s
[Pipeline] sh
```

Figure 4-14 Sample Output



4.2 Running OCNADD Test Cases using ATS

This section describes how to run Oracle Communications Network Analytics Data Director (OCNADD) test cases using ATS. It includes:

- Prerequisites
- Logging in to ATS
- OCNADD-NewFeatures Pipeline
- OCNADD-NewFeatures Documentation
- Troubleshooting ATS

4.2.1 Prerequisites

To run OCNADD test cases, ensure that the following prerequisites are met:

- The ATS version must be compatible with the OCNADD release.
- The ATS and Stub must be deployed in the namespace based on the OCANDD deployment mode, see Note.
- For OCANDD deployments in which ACL is enabled, perform the following steps:

> **Note**
>
> Skip these steps if ACL is not enabled in the OCNADD deployment.

1. Set the Jenkins pipeline variables *INTRA_TLS_ENABLED* and *ACL_ENABLED* to *true*.
2. Update the *ssl.truststore.password, ssl.keystore.password,* and *ssl.key.password* in *ocnadd_tests/data/admin.properties* file inside the ATS pod as follows:
   a. Access the Kafka pod from the OCNADD worker group or the default group deployment in which the SCRAM user configuration was added while enabling the

ACL. Run the following command, in this example `kafka-broker-0` is the Kafka pod:

```
kubectl exec -it kafka-broker-0 -n <namespace> -- bash
```

b. Extract the *ssl* parameters from the Kafka broker environments, run the following command:

```
env | grep -i pass
```

c. Use the *truststore* and *keystore* passwords retrieved from the above command output to update the *ocnadd_tests/data/admin.properties* file of the ATS pod, run the following commands:

```
 kubectl exec -it ocats-ocats-ocnadd-xxxx-xxxx -n <namespace> --
bash


$ vi ocnadd_tests/data/admin.properties
```

Sample output:

```
              security.protocol=SASL_SSL
              sasl.mechanism=PLAIN

sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginMo
dule required username="ocnadd" password="ocnadd";
              ssl.truststore.location=/var/securityfiles/keystore/
trustStore.p12
              ssl.truststore.password=<truststore pass>
              ssl.keystore.location=/var/securityfiles/keystore/
keyStore.p12
              ssl.keystore.password=<keystore pass>
              ssl.key.password=<keystore pass>
```

Note

- Before triggering a new pipeline from Jenkins, delete all existing Data Feed and Filter configurations from the previous execution.
- If ATS and Stub are in OCNADD centralized deployment mode, exclude the scenario tag *NotSupportedForCentralizedDD* follow the step below:
  - In the Jenkins UI, navigate from **FilterWithTags(Yes)** to **Scenario_Exclude_Tags**, enter **NotSupportedForCentralizedDD** and click **Submit**.
    This step avoids failures from the various scenarios that are not supported by the OCNADD in the centralized deployment mode.

4.2.2 Logging in to ATS

Before logging in to ATS, you need to ensure that ATS is deployed successfully using HELM charts as shown below:

```
[ocnadd@k8s-bastion ~]$ helm status ocats -n ocnadd-deploy
NAME: ocats
LAST DEPLOYED: Sat Nov  3 03:48:27 2022
NAMESPACE: ocnadd-deploy
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
# Copyright 2018 (C), Oracle and/or its affiliates. All rights reserved.

Thank you for installing ocats-ocnadd.

Your release is named ocats , Release Revision: 1.
To learn more about the release, try:

  $ helm status ocats
  $ helm get ocats


[ocnadd@k8s-bastion ~]$ kubectl get pod -n ocnadd-deploy | grep ocats
ocats-ocats-ocnadd-54ffddb548-4j8cx         1/1     Running     0          9h

[ocnadd@k8s-bastion ~]$ kubectl get svc -n ocnadd-deploy | grep ocats
ocats-ocats-ocnadd          LoadBalancer   10.20.30.40    <pending>
8080:12345/TCP       9h
```

For more information on verifying ATS deployment, see Verifying ATS Deployment.

To log in to ATS, open a browser and provide the IP Address and port details as <Worker-Node-IP>:<Node-Port-of-ATS>.

> **Note**
>
> If LoadBalancer IP is provided, then give <LoadBalancer IP>:8080

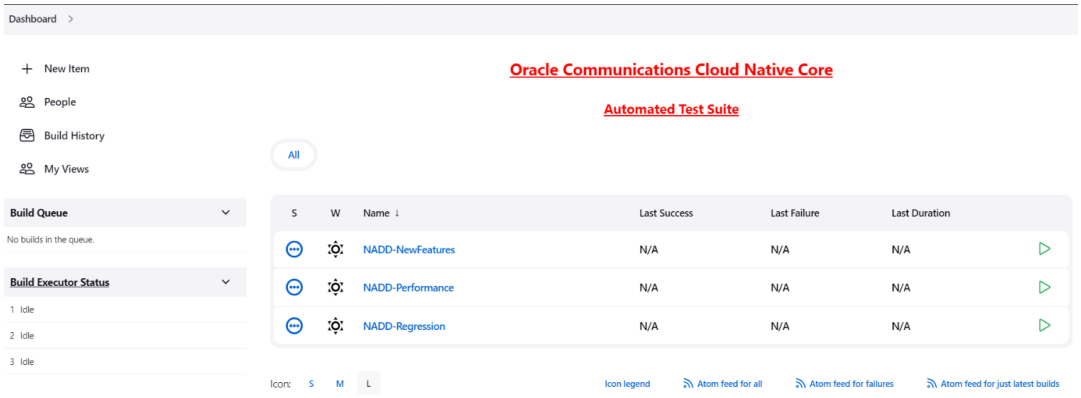
Figure 4-15 ATS Login

Running ATS

To run ATS:

1. Enter the login credentials. Click **Sign in**. The following screen appears.

Figure 4-16 OCNADD Pre-Configured Pipelines



OCNADD ATS has three pre-configured pipelines.

- **OCNADD-NewFeatures:** This pipeline has all the test cases delivered as part of OCNADD ATS - 23.4.0.
- **OCNADD-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
- **OCNADD-Regression:** This pipeline is not operational as of now. It is reserved for future releases of ATS.

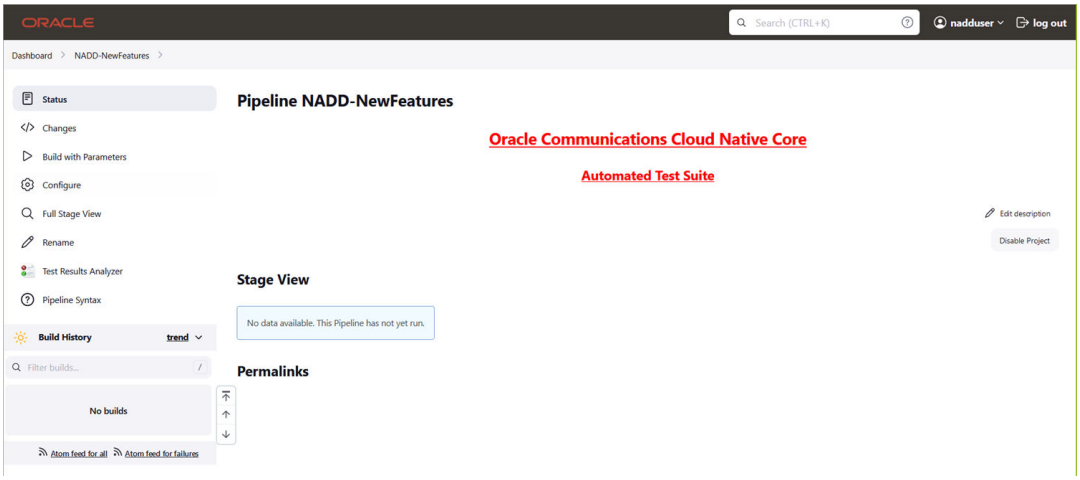
4.2.3 OCNADD-NewFeatures Pipeline

OCNADD-NewFeatures Pipeline

This is a pre-configured pipeline where users can run all the OCNADD new test cases. To configure its parameters, which is a one time activity, perform the following steps:

1. Click **OCNADD-NewFeatures** in the **Name** column. The following screen appears:

Figure 4-17 Configuring OCNADD-New Features



In the above screen:

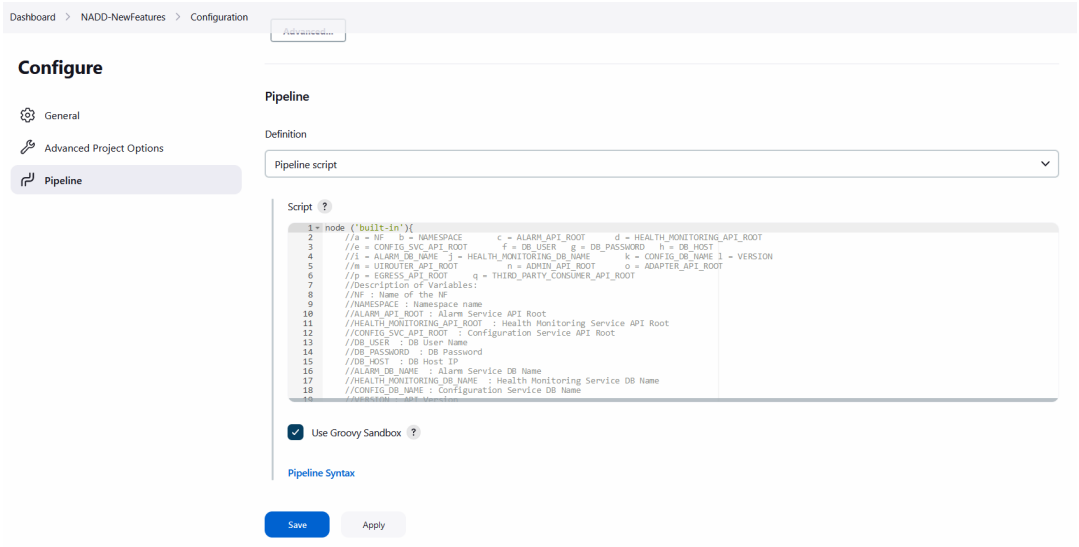
- Click **Configure** to configure OCNADD-New Features.
- Click **Build History** box to view all the previous pipeline executions, and the Console Output of each execution.
- The **Stage View** represents the previously run pipelines for reference.
- The **Test Results Analyzer** is the plugin integrated into the OCNADD-ATS. This option can be used to display the build-wise history of all the tests. It provides a consolidated graphical representation of all past tests.

2. Click **Configure**. Once the page loads completely, click the **Pipeline** tab:

> **Note**
>
> Make sure that the **Configure** page loads completely before you perform any action on it. Also, do not modify any configuration other than shown below.

The **Pipeline** section of the configuration page appears as follows:

Figure 4-18 Pipeline Section



Important

Remove the existing default content of the pipeline script and copy the following script content.

The content of the pipeline script is as follows:

```
node ('built-in'){
    //a = NF    b = NAMESPACE    c = DB_HOST    d = DB_USER    e =
DB_PASSWORD
    //f = ALARM_DB_NAME    g = HEALTH_MONITORING_DB_NAME    h =
CONFIG_DB_NAME
    //i = ALARM_API_ROOT    j = HEALTH_MONITORING_API_ROOT    k =
CONFIG_SVC_API_ROOT
    //l = UIROUTER_API_ROOT    m = ADMIN_API_ROOT    n =
THIRD_PARTY_CONSUMER_API_ROOT
    //o = BACKUP_RESTORE_IMG_PATH    p = ALERT_MANAGER_URI      q =
PROMETHEUS_URI      r = INTRA_TLS_ENABLED
    //s = RERUN_COUNT     t = ACL_ENABLED     u = WORKER_GROUP    v =
MANAGEMENT_GROUP
    //Description of Variables:
    //NF : Name of the NF
    //NAMESPACE : Namespace name
    //DB_HOST : DB Host IP
    //DB_USER : DB User Name
    //DB_PASSWORD : DB Password
    //ALARM_DB_NAME : Alarm Service DB Name
    //HEALTH_MONITORING_DB_NAME : Health Monitoring Service DB Name
    //CONFIG_DB_NAME : Configuration Service DB Name
    //ALARM_API_ROOT : Alarm Service API Root
    //HEALTH_MONITORING_API_ROOT : Health Monitoring Service API Root
    //CONFIG_SVC_API_ROOT : Configuration Service API Root
    //UIROUTER_API_ROOT : UI Router API Root
```

```
    //ADMIN_API_ROOT : Admin Service API Root
    //THIRD_PARTY_CONSUMER_API_ROOT : Third Pary Consumer API Root
    //BACKUP_RESTORE_IMG_PATH : Repository path for Backup restore image
    //ALERT_MANAGER_URI : Alert Manager API Root
    //PROMETHEUS_URI : Prometheus URI
    //INTRA_TLS_ENABLED : IntraTLS Value true/false
    //RERUN_COUNT : ReRun Count for Failed Tests
    //ACL_ENABLED : ACL Enabled true/false
    //WORKER_GROUP : Worker Group Namespace name
    //MANAGEMENT_GROUP : Management Group Namespace name

    sh '''
        sh /var/lib/jenkins/ocnadd_tests/preTestConfig-NewFeatures-NADD.sh
\
        -a NADD \
        -b <ocnadd-namespace> \
        -c <DB_HOST> \
        -d <DB_USER> \
        -e <DB_PASSWORD> \
        -f alarm_schema \
        -g healthdb_schema \
        -h configuration_schema \
        -i ocnaddalarm.<management-group-namespace>.svc.<domainName>:9099 \
        -j ocnaddhealthmonitoring.<management-group-
namespace>.svc.<domainName>:12591 \
        -k ocnaddconfiguration.<management-group-
namespace>.svc.<domainName>:12590 \
        -l ocnaddbackendrouter.<management-group-
namespace>.svc.<domainName>:8988 \
        -m ocnaddadminservice.<management-group-
namespace>.svc.<domainName>:9181 \
        -n ocnaddthirdpartyconsumer.<worker-group-
namespace>.svc.<domainName>:9094 \
        -o <repo-path>/ocdd.repo/ocnaddbackuprestore:<tag> \
        -p occne-kube-prom-stack-kube-alertmanager.occne-
infra.svc.<domainName>:80/<clusterName> \
        -q occne-kube-prom-stack-kube-prometheus.occne-
infra.svc.<domainName>:80/<clusterName>/prometheus/api/v1/query \
        -r true \
        -s 2 \
        -t false \
        -u <worker-group-namespace:clusterName> \
        -v <management-group-namespace> \
    '''
    if(env.Include_Regression && "${Include_Regression}" == "YES"){
        sh '''sh /var/lib/jenkins/common_scripts/merge_jenkinsfile.sh'''
        load "/var/lib/jenkins/ocnadd_tests/jenkinsData/Jenkinsfile-NADD-
Merged"
    }
    else{
        load "/var/lib/jenkins/ocnadd_tests/jenkinsData/Jenkinsfile-NADD-
NewFeatures"
    }

}
```

You can modify pipeline script parameters from **"-b"** to **"-q"** on the basis of your deployment environment, click on 'Save' after making the necessary changes.

The description of all the script parameters is as follows:

- **a**: Name of the NF to be tested in the capital (NADD).
- **b**: Namespace in which the NADD is deployed (ocnadd-deploy).
- **c**: DB Host IP provided NADD deployment (10.XX.XX.XX).
- **d**: DB username provided during NADD deployment.
- **e**: DB password provided during NADD deployment.
- **f**: DB Schema Name of ocnaddalarm microservice provided during NADD deployment (alarm_schema).
- **g**: DB Schema Name of ocnaddhealthmonitoring microservice provided during NADD deployment (healthdb_schema).
- **h**: DB Schema Name of ocnaddconfiguration microservice provided during NADD deployment (configuration_schema).
- **i**: API root endpoint to reach NADD's ocnaddalarm microservice. (<Worker-Node-IP>:<Node-Port-of-ocnaddalarm>) or default value
- **j**: API root endpoint to reach NADD's ocnaddhealthmonitoring microservice. (<Worker-Node-IP>:<Node-Port-of-ocnaddhealthmonitoring>) or default value
- **k**: API root endpoint to reach NADD's ocnaddconfiguration microservice. (<Worker-Node-IP>:<Node-Port-of-ocnaddconfiguration>) or default value
- **l**: API root endpoint to reach NADD's ocnaddbackendrouter microservice. (Not used in the current release, use the default value)
- **m**: API root endpoint to reach NADD's ocnaddadminservice microservice. (Not used in the current release, use the default value)
- **n**: API root endpoint to reach NADD's ocnaddthirdpartyconsumer microservice. (<Worker-Node-IP>:<Node-Port-of-ocnaddthirdpartyconsumer>) or default value
- **o**: Repository path for ocnaddbackuprestore image.
- **p**: API root endpoint to reach alert manager microservice.
- **q**: Prometheus URI
- **r**: Set the IntraTLS value to either "true" or "false" based on user requirement and OCNADD deployment (either with intraTlS enabled or disabled).
- **s**: Rerun count for failed test cases. The default value is set to '2'. It can be set to '0', '1', or any other value based on user requirements.
- **t**: Set the ACL_ENABLED value to either "true" or "false" based on the OCNADD deployment (ACL enabled or disabled).
- **u**: Worker group namespace name along with cluster name.
- **v**: Management group namespace name.

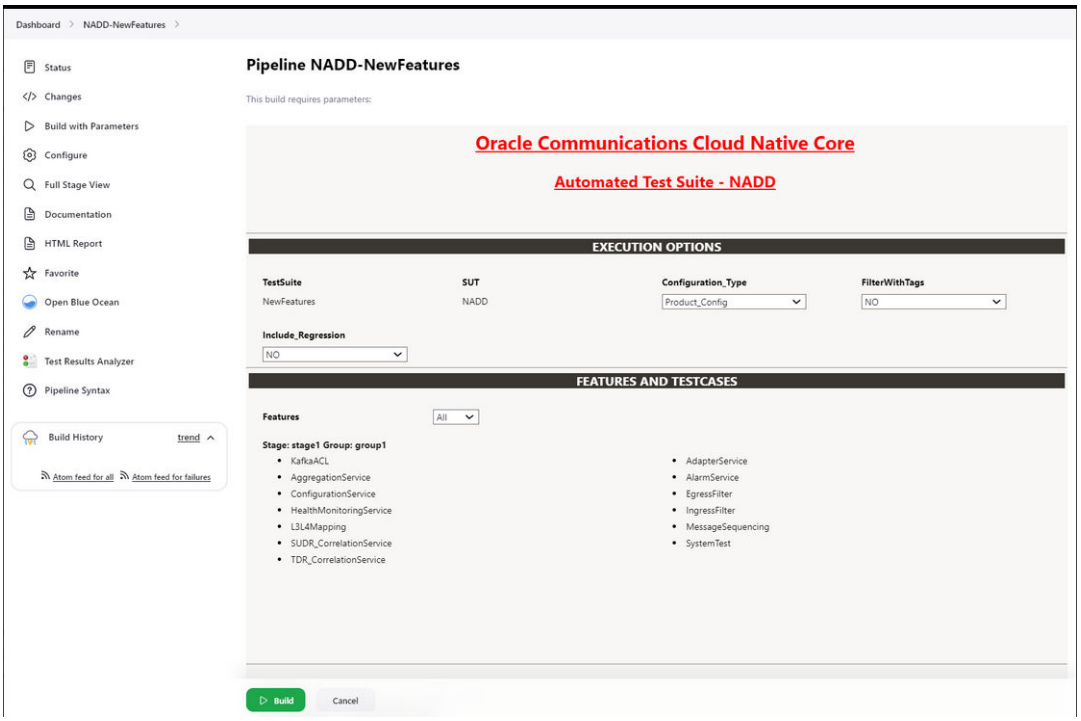
> **Note**
>
> If ATS and Stub are on a OCNADD in non-centralized deployment mode, then u: <worker-group-namespace:clusterName> and v: <management-group-namespace> will be in the same namespace.

Running OCNADD Test Cases

To run OCNADD test cases, perform the following steps:

1. Click the **Build with Parameters** link available in the left navigation pane of the **NADD-NewFeatures** Pipeline screen. The following page appears:

Figure 4-19 Pipeline NADD_NewFeatures



2. Select **Configuration_Type** as **Product_Config**.
3. Set **Include_Regression** to 'NO' from the drop-down list.
4. In **Select_Option**:
   - Select **All** to run all the feature test cases and click the **Build** button to run the pipeline.
   - Choose **Single/MultipleFeatures** to run the specific feature test cases and click the **Build** button to run the pipeline.
   - Choose **MultipleFeature/MultipleTestCases** to select multiple features and multiple test cases within the selected features and click the **Build** button to run the pipeline.

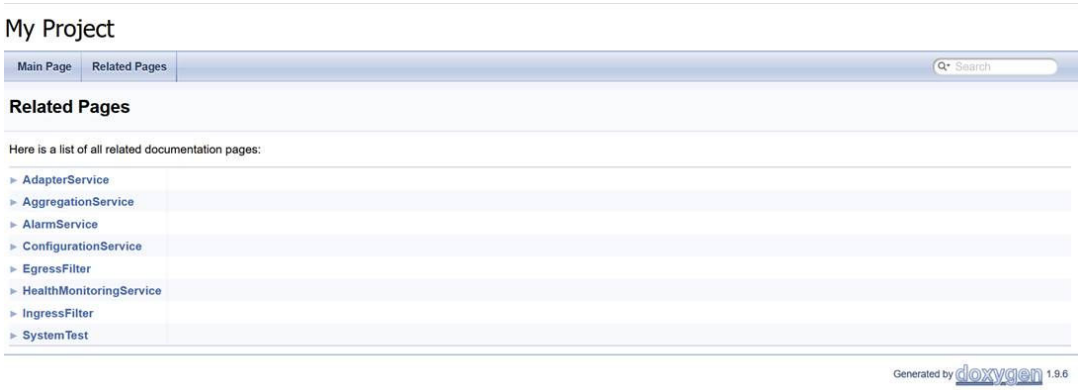
4.2.4 OCNADD-NewFeatures Documentation

The NADD-NewFeatures pipeline has a HTML report of all the feature files that can be tested as part of the OCNADD ATS release. Follow the procedure listed below to view all the OCNADD functionalities:

1. On the **Pipeline NADD-NewFeatures** page click **Documentation** in the left navigation pane. All test cases provided as part of the OCNADD ATS release are displayed on the screen.

> **Note**
>
> - The **Documentation** option appears on the screen only if NADD-NewFeatures pipeline test cases are executed at least once.
> - Use the Firefox browser to open the **Documentation**, other browsers are not supported.

Figure 4-20 Documentation



2. Click any functionality to view its test cases and scenarios for each test case.
3. To exit, click **Back to NADD-NewFeatures** on the top-left corner of the screen.

4.2.5 Troubleshooting ATS

This section provides troubleshooting procedures for some common ATS test case failures.

Offset Count Mismatch Results in Test Case Failure

Problem

Test cases fail due to the offset count mismatch. Delay in third-party consumers receiving messages results in this failure.

Sample Error Message

```
Example: The test case failed due to offset count of the MAIN topic(which is
186) does not match with off set count of the third-party Consumer(which is
155)
Then Compare the offset change in MAIN topic and consumer ... failed in 0.000s
Assertion Failed: The increase in offset counts is not matching, increase in
offset of MAIN : 186 , increase in offset of Consumer : 155
Captured stdout:
{'Location': 'http://ocnaddconfiguration:12590/ocnadd-configuration/
configure/v3/app-oracle-cipher', 'content-length': '0'}
['OCL', '2023-08-13T19:49:49.944Z', 'INFO', '1', '---', '[', 'scheduling-1]',
'c', '.o', '.c', '.c', '.o', '.c', '.c', '.ConsumerController', ':', '|',
'*TOTAL*', '|', '0', '(0', ')', '|']
['OCL', '2023-08-13T19:49:49.944Z', 'INFO', '1', '---', '[', 'scheduling-1]',
'c', '.o', '.c', '.c', '.o', '.c', '.c', '.ConsumerController', ':',
'+------------------------------+------------------------------
```

```
+------------------------------+------------------------------------+']
['OCL', '2023-08-13T19:50:14.952Z', 'INFO', '1', '---', '[', 'scheduling-1]',
'c', '.o', '.c', '.c', '.o', '.c', '.c', '.ConsumerController', ':', '|',
'*TOTAL*', '|', '155', '(0', ')', '|']
['OCL', '2023-08-13T19:50:14.952Z', 'INFO', '1', '---', '[', 'scheduling-1]',
'c', '.o', '.c', '.c', '.o', '.c', '.c', '.ConsumerController', ':',
'+------------------------------+------------------------------
+------------------------------+------------------------------------+']
```

Solution

The test cases which failed due to offset count mismatch, pass when a test case rerun is performed.

Assertion Failed: Status code:404 did not match with 204

Problem

The test cases fail due to 'Assertion Failed: Status code:404 did not match with 204'. This error occurs when any previous scenario abruptly fails without executing all the steps of the scenario.

Sample Error Message

```
Example:
Given delete already existing data feeds ... failed in 0.346s
Assertion Failed: FAILED SUB-STEP: Given delete already existing
configurations
Substep info: Assertion Failed: Status code:404 did not match with 204
Traceback (of failed substep):
  File "/env/lib64/python3.9/site-packages/behave/model.py", line 1329, in run
    match.run(runner.context)
  File "/env/lib64/python3.9/site-packages/behave/matchers.py", line 98, in
run
    self.func(context, *args, **kwargs)
  File "/var/lib/jenkins/cncats/ocnftest/ocnadd_steps.py", line 1906, in
step_impl
    assert False , 'Status code:{} did not match with
204'.format(context.response.status_code)
```

Solution

The test cases which failed due to previous scenario failures, pass when a test case rerun is performed.

OCNADD Pods Enter a 'Image Pull' Error State

Problem

The OCNADD pods enter a 'Image Pull' error state after a few test cases are executed. This occurs when a test case appends a suffix to the image name and its execution is incomplete.

Solution

Correct the image name by editing the pods deployment and rerun the test suite.