Oracle® Communications

Network Analytics Automated Testing Suite Guide

Release 23.1.0
F78251-01
March 2023

Oracle Communications Network Analytics Automated Testing Suite Guide, Release 23.1.0

F78251-01

Contents

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 (CNE)
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 23.1.0 in *Oracle Communications Network Analytics Automated Testing Suite Guide*.

Release 23.1.0 - F78251-01, March 2023

OCNADD Release 23.1.0

- Updated the ATS Framework Features section to outline the OCNADD support for ATS features in this release.
- Updated the Resource Requirements section to modify the overall resource usage in terms of CPUs and memory for OCNADD.
- Updated the OCNADD Pods resource requirements details in Resource Requirements section.
- Changed the package format to .tgz file in Downloading the ATS Package section.
- Updated Pushing the Images to Customer Docker Registry section to correct the various filenames.
- Updated Verifying ATS Deployment section to update the command name for ATS verification.
- Updated the Note in Deploying ATS and Stub in Kubernetes Cluster section for running ATS Suit.
- Updated the pipeline script and description of script parameters in OCNADD-NewFeatures Pipeline section.

OCNWDAF Release 23.1.0

- Updated the ATS Framework Features section to outline the OCNWDAF support for ATS features in this release.
- Updated the CNE version in the Software Requirements section.
- Updated the Downloading the ATS Package section to include the updated package structure.
- Updated the OCNWDAF and ATS release numbers in the Pushing the Images to Customer Docker Registry section.
- Updated the Deploying NWDAF ATS in the Kubernetes Cluster section with latest sample image.
- Updated the commands in Verifying ATS Deployment section.
- Updated the Running NWDAF Test Cases using ATS section with steps to configure the pipeline.

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

Through Automated Testing Suite (ATS), Oracle Communications provides an end-to-end solution to its customers for deploying and testing its Network Analytics Suite products.

This document describes the implementation of ATS for the following products:

- Network Data Analytics Function (OCNWDAF)
- Network Analytics Data Director (OCNADD)

ATS for Network Analytics Suite

For Network Analytics Suite NFs (Network Functions), ATS is built using Oracle Linux 8-slim as the base image. Jenkins is a part of the ATS image and it provides a GUI interface to the users to test either a single NF or multiple NFs independently in the same environment.

Along with the NF docker images, users are provided with the ATS image, simulator images, and test cases of the specific NF. All these images and test cases together form a fully automated suite using which users can directly deploy and test. You can combine it with any other Continuous Integration (CI) pipeline with minimal changes, as Network Analytics Suite ATS uses Jenkins as GUI.

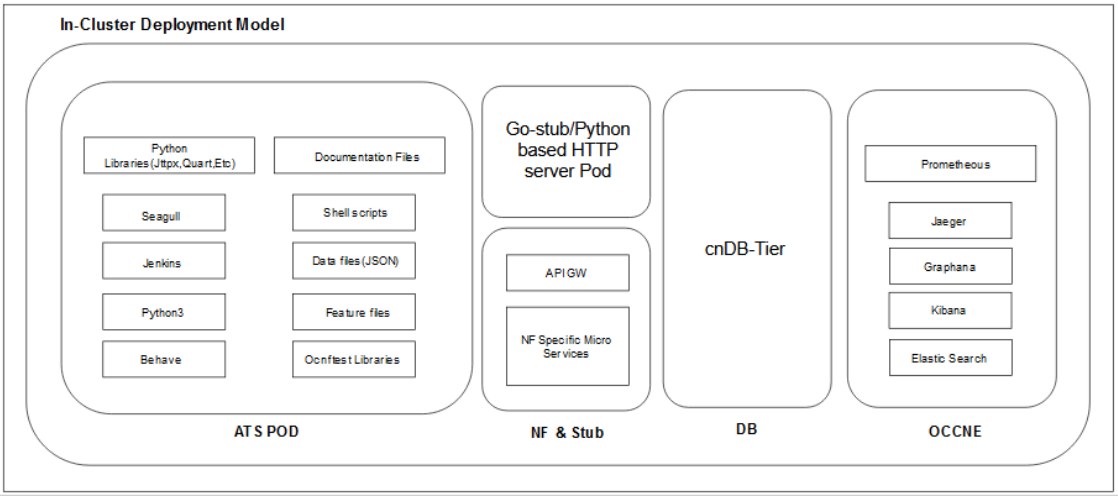
Automated Testing Suite (ATS):

- Provides an end-to-end solution to the customers for testing Oracle Communications Network Analytics Suite -NFs. The ATS package includes:
  - 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 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).
- Enables all the NF teams with basic environment, framework, and a GUI (Jenkins) to run all the functional test cases.

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 any network function, refer the following documents available on Network Analytics Suite Securesites.

- *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
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	No

2.1 Application Log Collection

Application Log Collection helps with debugging if a test case fails by collecting the application logs for NF System Under Test (SUT). Application logs are collected for the duration the failing test case was executed.

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

Application Log Collection Using ElasticSearch

User Options

- Fetch_Log_Upon_Failure: YES/NO to select whether the log collection is required for a particular run
- Log_Level: NF Log-Levels to set the different Log-Level available for microservice

Fetching Logs

- ElasticSearch api is used to access & fetch logs
- Logs are fetched from ElasticSearch for the failed scenarios
- Hooks (after scenario) within the cleanup file are used to trigger
- Duration of the failed scenario is calculated based on the time stamp and is passed as a parameter to fetch the logs from ElasticSearch
- Filtered query is used to fetch the records based on Pod name, Service name & timestamp (Failed Scenario Duration)
- No rollover or rotation of logs

Considerations

- Maximum records that ElasticSearch API can fetch per microservice for a failed scenario is limited to 10k.

Versions

- ElasticSearch: 7.8.0
- python-ElasticSearch: 7.12.1

Parameters

- ElasticSearch: Kubernetes latency duration is considered as External parameter which is configurable in Jenkins
- ElasticSearch: Host name and port

Application Log Collection Using Kubernetes Logs

User Options

- Fetch_Log_Upon_Failure: YES / NO to select whether Log Collection is required for a particular run
- Log_Level: NF Log-Levels to set the different Log-Level available for Micro-Service

Fetching Logs

- Kube API is used to access and fetch logs
- Logs are fetched from microservices directly for the failed scenarios
- Hooks (after scenario) within the cleanup file are used to trigger
- Duration of the failed scenario is calculated based on the time stamp and is passed as a parameter to fetch the logs from microservices

Considerations

- Logs roll over can happen while fetching the logs for a failed scenario; max loss of logs are confined to a single scenario only.

2.2 ATS API

The Application Programming Interface (API) feature offers APIs to perform routine ATS tasks as described in the following:

- Start: To initiate one of the three test suites such as Regression, NewFeatures 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

Prerequisites

- Create an apiuser and grant the required access
- Create an API token for authentication in API calls

Create an API User

Perform the following procedure to create an API user:

1. Log in to the ATS application using admin credentials.
2. Click **Manage ATS** in the left navigation pane of the ATS application.
3. Scroll down and click **Manage Users**.
4. Click **Create User** in the left navigation pane.
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.

Grant Access to the API User

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

1. Click **Manage ATS** in the left navigation pane.
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. Click **Configure** in the left navigation pane.
9. Scroll down to **Enable project-based security** and click **Add user.**
10. Enter the **username** created in the prompt that appears.
11. Check all the boxes in the Authorization matrix for apiuser that are also checked for `<nf>user`.
12. Click **Save**. Now, apiuser can be used in API calls.

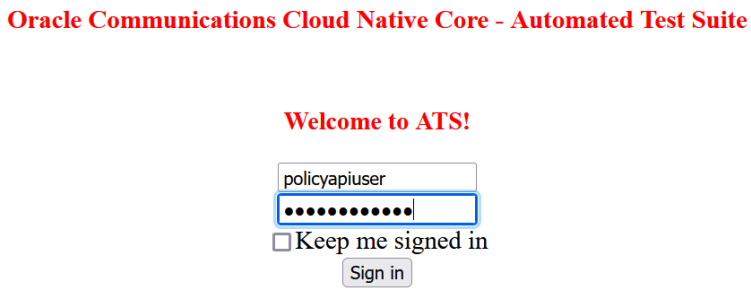
Generate an API Token for a User

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:

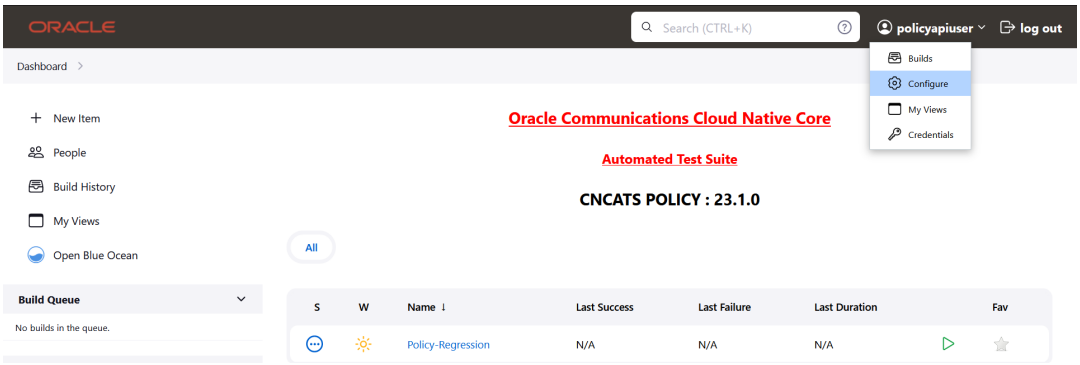
1. Log in to Jenkins as an NF apiuser to generate an API token:

Figure 2-1 ATS Login Page



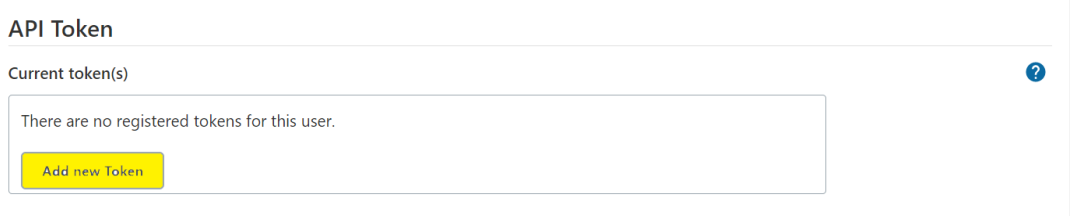
2. Click on username from the drop-down list at the top right of Jenkins GUI, then click **Configure**:

Figure 2-2 Configure to Add Token



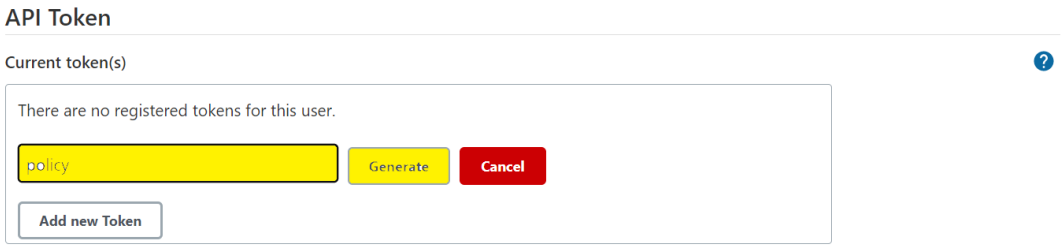
3. Under the **API Token** section, click **Add New Token**:

Figure 2-3 Add New Token



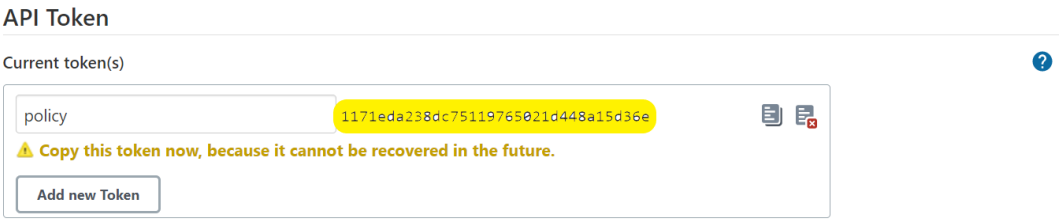
4. Enter a suitable name for the token, such as policy, and then click **Generate**:

Figure 2-4 Generate Token



5. Copy the generated token that appears and save it. You will not be able to retrieve the token once you close this 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.

2.3 ATS Health Check

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, this process has now been automated.

On clicking **Build Now**, the user can run the health check on ATS and store its results in the console logs.

Deploying ATS Health Check in a Webscale Environment

Deploy ATS health check with `Webscale` parameter set to 'true' and following parameters by encoding it with base64 in the ATS values.yaml file:

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

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 a 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 UDR it is called as UDR-HealthCheck pipeline.

3. Provide parameter a with Helm release name deployed. If there are multiple releases, use comma to provide all Helm release names.
   Provide parameter c with Helm command name- helm or helm3, or helm2, whichever works.

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

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

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

Deploying ATS Health Check in a Non-Webscale Environment

Deploy ATS health check with `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 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**.
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 Helm command name- helm or helm3, or helm2, whichever works.

   ```
   //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]
   ```

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

2.4 ATS Jenkins Job Queue

The ATS Jenkins Job Queue feature is to queue the second job if the current job is already running, whether from the same pipelines or different pipelines.

1. In Jenkins configuration, set the total number of executors to 1.
   This makes the jobs wait for resource allocation if the new pipeline is triggered.

   > **Note**
   >
   > This change can be done in the base image.

2. Change the agent type to **none**.

```
pipeline {
    agent none
    stages {
        stage ('Preparation') {
            steps {
                script {
                    echo "Here the health check will be performed for ATS in future releases."
                    testop = "${TestSuite}"
                    selectop = "${Select_Option}"
                    echo testop
                    echo selectop

                }
            }
        }

        stage ('Execute-Tests') {
            steps {
                catchError(buildResult: 'SUCCESS', stageResult: 'UNSTABLE') {
                    script{
                        try {
```

3. Remove the new node allocation from the post-build action within the Jenkins pipeline script.

```
post {
    always {
        // node ('master'){
            script {
            echo "If any cleanup is required, it will be done in this stage for future release."
                if("${SUTSuite}" == "ALL")
                {
                    publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: true, reportDir:
                    '/var/lib/jenkins/ocudr_tests/documentation/Regression/ALL/html', reportFiles: 'pages.html', reportName:
                    'Documentation', reportTitles: ''])
                    sh '''
                    cp -r  /var/lib/jenkins/ocudr_tests/reports/ $WORKSPACE/test-results
                    rm -rf /var/lib/jenkins/ocudr_tests/reports/*
                    '''
                    junit '**/test-results/**/*.xml'
                    }
                else if("${SUTSuite}" == "PCF")
                {
                    publishHTML([allowMissing: false, alwaysLinkToLastBuild: false, keepAll: true, reportDir:
                    '/var/lib/jenkins/ocudr_tests/documentation/Regression/PCF/html', reportFiles: 'pages.html', reportName:
                    'Documentation', reportTitles: ''])
```

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
```

ATS Scripts

- **ats_backup.sh**: This script requires the user's inputs 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 just the Jenkins jobs and user's folder or just 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 inputs and uninstalls the corresponding ATS.

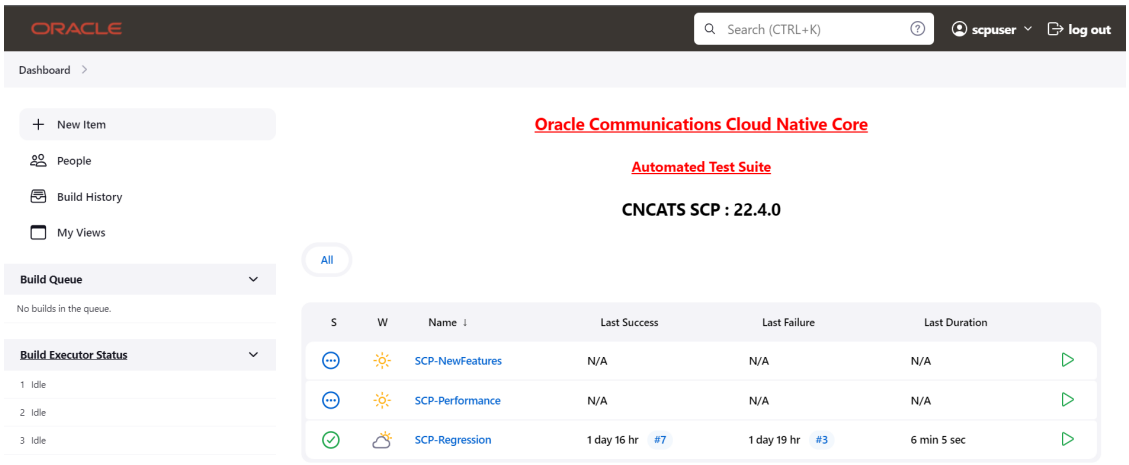
- **ats_install.sh**: This script requires the user's inputs and installs a new ATS. If `PVEnabled` is set to `true`, the script also reads the PVC name from `values.yaml` and creates the `values.yaml` before installation. Also, if needed, the script performs the postinstallation steps, such as copying 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 and restores the new release ATS pipeline and views the configuration by referring to the last release ATS Jenkins jobs and user's configuration. It depends on the user whether or not 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 user's configuration to restore.
  The script refers to the last release ATS Jenkins pipelines and sets the `Discard Old builds` property, provided that this property is set in the last release ATS for a pipeline but not in the current release. If this property is set in both the 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 ATS. If any custom pipeline (created by the user) is present in the last release ATS, the script restores that as well. It also restores the extra views created by NF users, for example, policyuser, scpuser, and nrfuser. 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) introduced in the new release.

  While deploying ATS without PV, Jenkins of ATS 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** of Jenkins for the restore process to complete.

2.6 ATS System Name and Version Display on Jenkins GUI

There is a `init.groovy` script present in the `.jenkins` folder in the ATS pod. The script reads the system name and version values from the ATS pod and displays them on the ATS GUI.

Figure 2-6 ATS System Name and Version

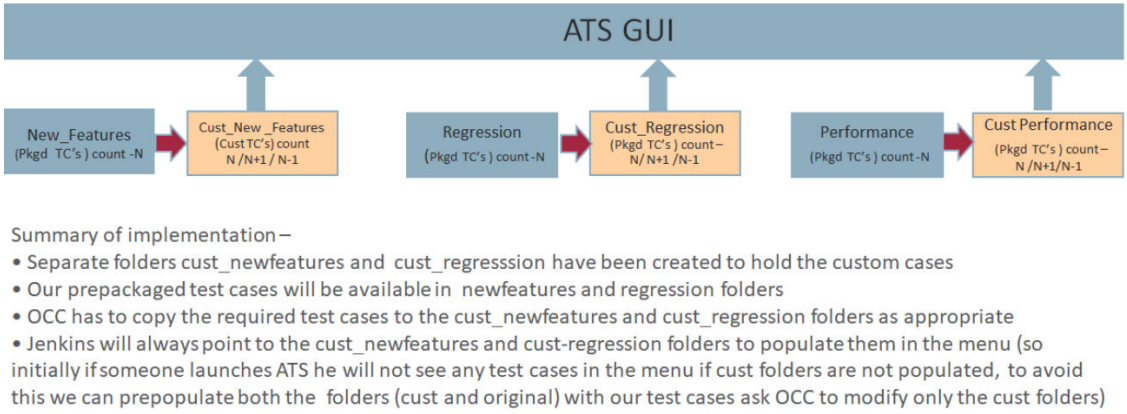


2.7 Custom Folder Implementation

With the implementation of custom folders (cust_newfeatures, cust_regression, and cust_performance), users can customize test cases (update, add, or delete test cases) without disturbing the original product test cases in the newfeatures, regression, and performance folders. The new customized test cases are stored in the custom folders.

Initially, the product test case folders and custom test case folders have same set of test cases. Subsequently, the users carry out customization in the custom test cases folders and ATS always runs the test cases from custom test cases folders.

Figure 2-7 Summary of Custom Folder Implementation



2.8 Single Click Job Creation

This feature enables ATS users to create a job to run TestSuite with single click.

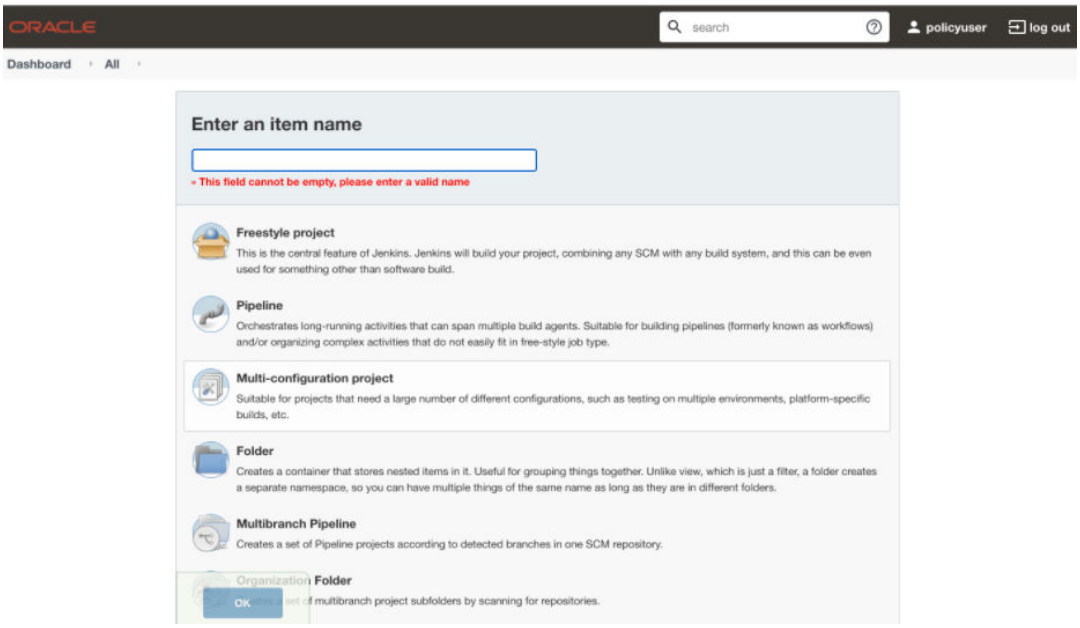
Prerequisite: The network function specific user should have the 'Create Job' access.

Configuring Single Click Job

To configure single click feature:

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

Figure 2-8 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.

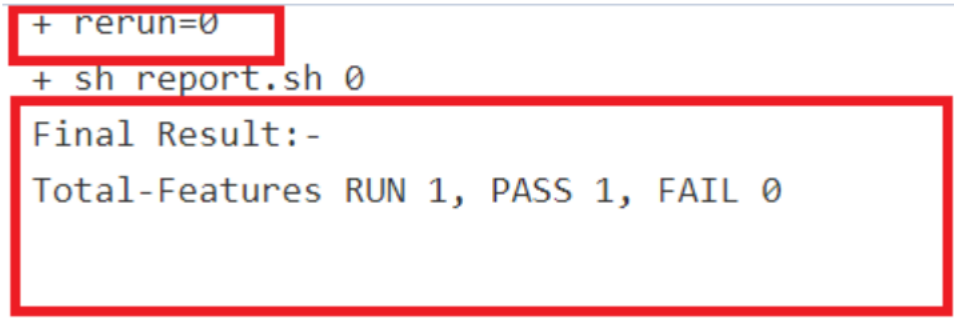
2.9 Final Summary Report, Build Color, and Application Log

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 of the final summary report displays the Total-Features output.

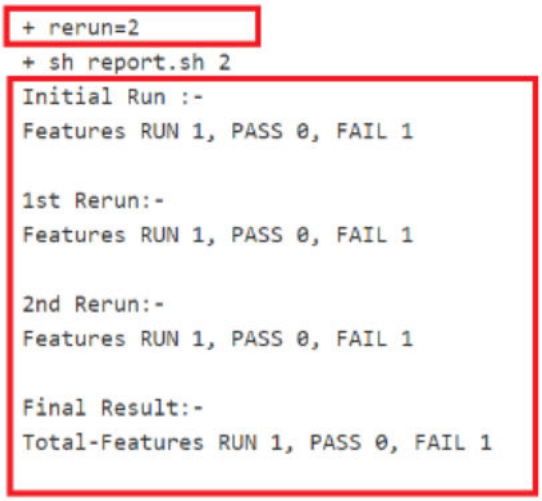
- If rerun is set as 0, the test result report shows the following result:

Figure 2-9 Total Features = 1, and Rerun = 0



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

Figure 2-10 Total Features = 1, and Rerun = 2



Changes After Integrating Parallel Test Execution Framework Feature

Final Summary Report Implementations

Figure 2-11 Group Wise Results

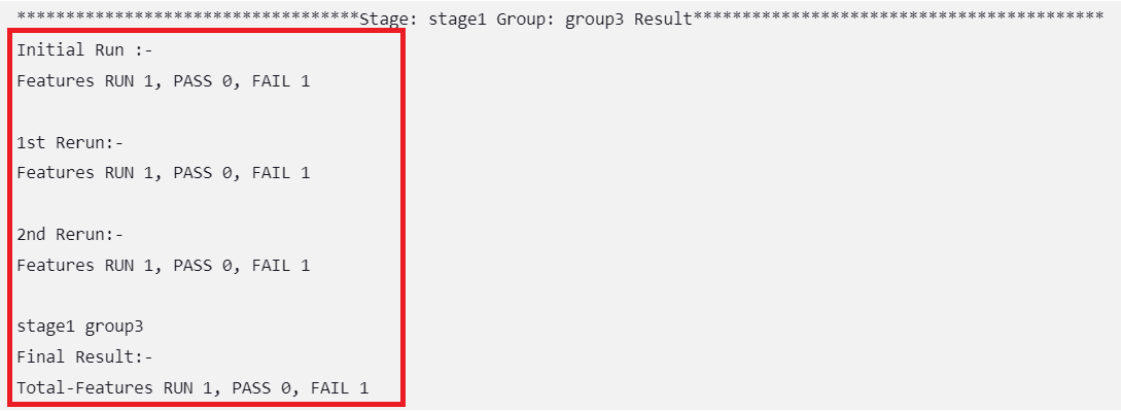


Figure 2-12 Result When Selected Features Pass

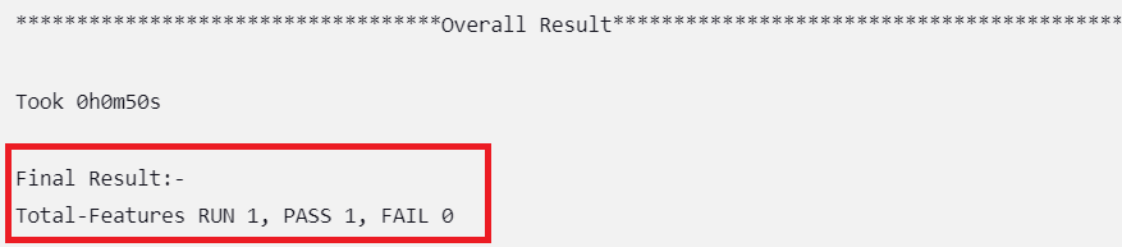


Figure 2-13 Result When Any of the Selected Features Fail

```
**********************************Overall Result****************************************
Failing scenarios:
Stage:stage1 Group:group3
  SCP_Registration_With_PLMNList.feature:45  Static configuration to NRF for all nfType

Took 0h1m59s

Initial Run:-
Features RUN 1, PASS 0, FAIL 1

1st Rerun:-
Featues RUN 1, PASS 0, FAIL 1

2nd Rerun:-
Featues RUN 1, PASS 0, FAIL 1

Final Result:-
Total-Featues RUN 1, PASS 0, FAIL 1
```

Implementing Build Colors

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

Table 2-2 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 two parameters: the rerun count and the pass or fail status of test cases in the initial and final runs.

From the parallel test case execution, the pipeline status will also depend 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 will be considered "NO"."

Table 2-3 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

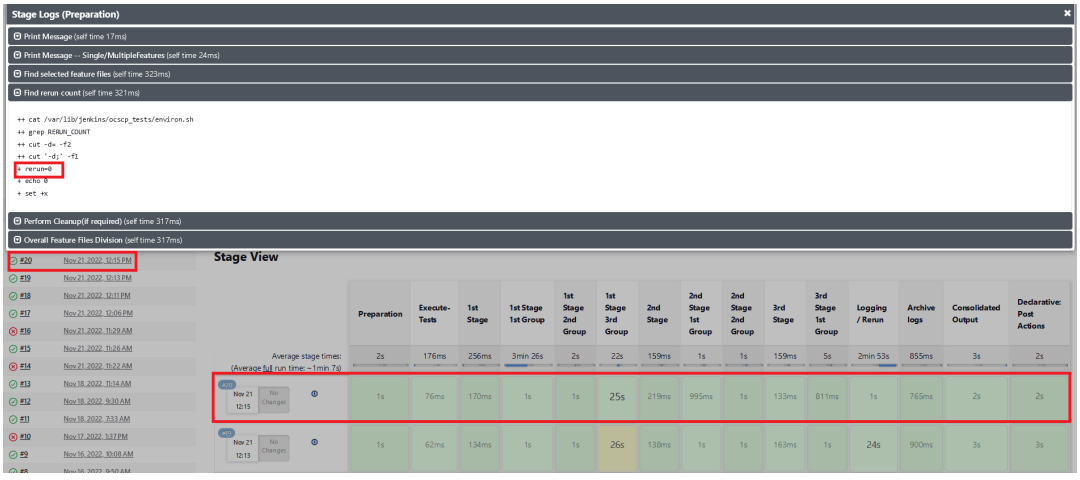
Table 2-4 Pipeline Status When Fetch_Log_Upon_Failure = YES

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:

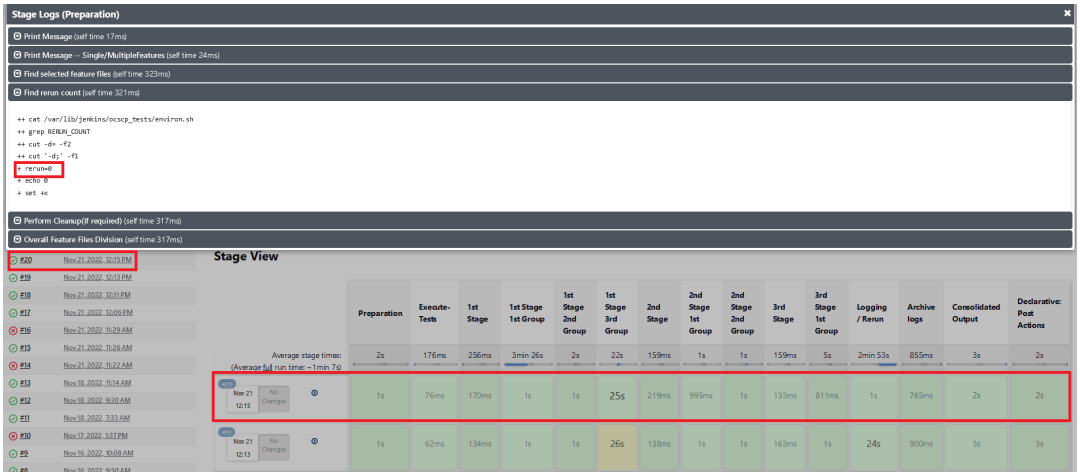
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases pass in the initial run. The pipeline will be green, and its status will show as blue:

Figure 2-14 Fetch_Log_Upon_Failure is set to YES and rerun_count is set to 0, test cases pass



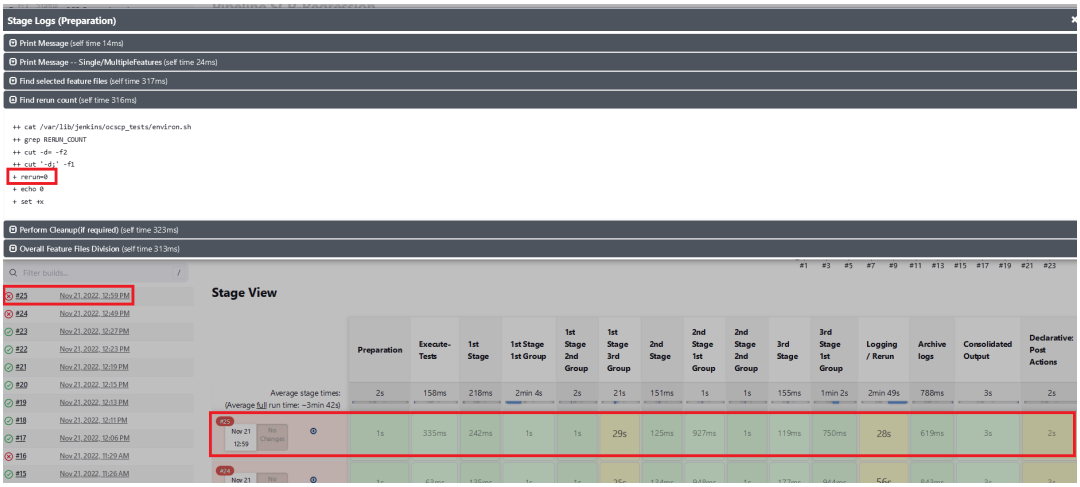
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases fail on the initial run but pass on the extra run. The initial execution stage will be yellow, all subsequent successful stages will be green, and the status will be blue:

Figure 2-15 Test Cases Fail on the Initial Run but Pass on the Extra Rerun



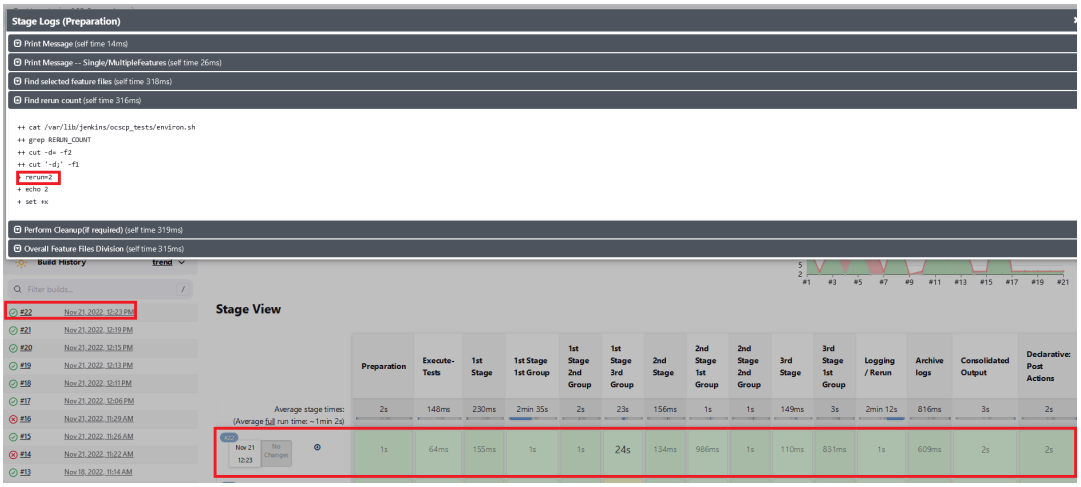
- When `Fetch_Log_Upon_Failure` is set to YES and `rerun_count` is set to 0, test cases fail in both the initial and the extra rerun. Execution stages will show as yellow, all other successful stages will be shown as green, and the overall pipeline status will be red:

Figure 2-16 Test Cases Fail in Both the initial and the Extra Rerun



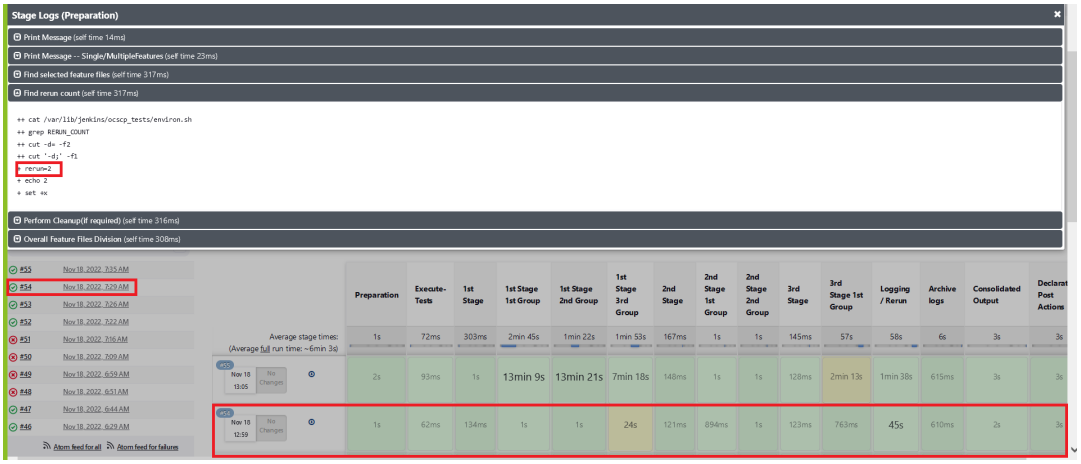
- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non zero. If all of the test cases pass in the first run, no rerun will be initiated because the cases have already been passed. The pipeline will be green, and the status will be indicated in blue:

Figure 2-17 All of the Test cases Pass in the Initial Run



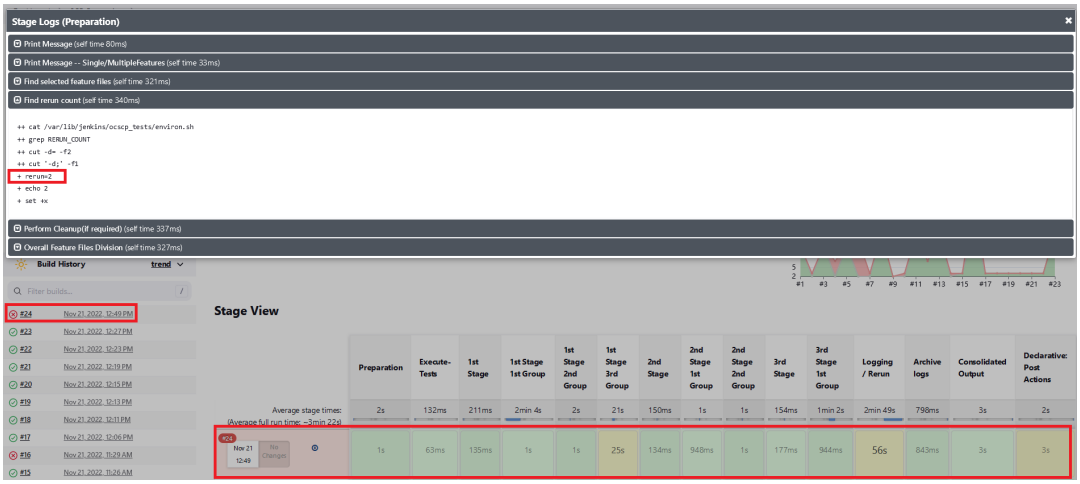
- When `Fetch_Log_Upon_Failure` is set to YES and the `rerun count` is set to non zero. If some of the test cases fail in the initial run and the remaining ones pass in one of the remaining reruns, then the initial test case execution stages will show as yellow, the remaining stages as green, and the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

Figure 2-18 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 2-19 Test Cases Fail in the Initial and Remaining Reruns



Implementing Application Log

ATS automatically fetches the SUT Debug logs during the rerun cycle if it encounters any failure and saves them in the same location as that of 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. Hence, the logs are fetched only for the microservices that has an impact or are associated with the failed test cases.

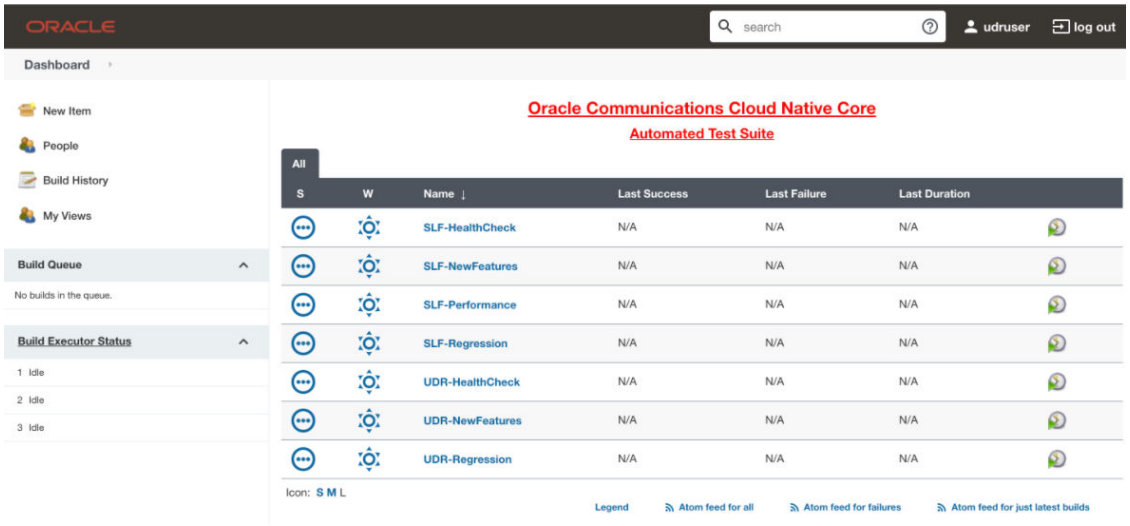
Location of SUT Logs: `/var/lib/jenkins/.jenkins/jobs/PARTICULAR-JOB-NAME/builds/BUILD-NUMBER/date-timestamp-BUILD-N.txt`

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

2.10 Lightweight Performance

With the implementation of Lightweight Performance feature in ATS, the ATS users can now run performance test cases. A new pipeline called <NF>-Performance (where, <NF> denotes the Network Function. For example, SLF-Performance is introduced in ATS.

Figure 2-20 Sample Screen: UDR Home Page



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

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

Configuring <NF>-Performance Pipeline

To configure <NF>-Performance:

1. Click **<NF>-Performance** pipeline and then, click **Configure**.
2. The **General** tab appears. The user must wait for the page to load completely.
3. Click the **Advanced Project Options** tab. Scroll-down to reach the **Pipeline** configuration section.
4. Update the configurations as per your NF requirements and click **Save**. The **Pipeline <NF>-Performance** page appears.
5. Click **Build Now**. This triggers lightweight traffic for respective network function. For more information see, #unique_48.

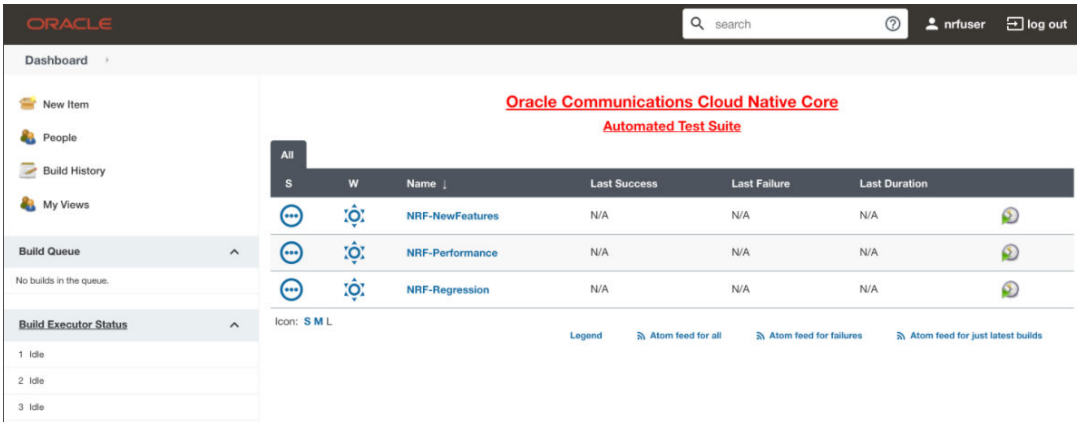
2.11 Modifying Login Password

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

To modify the default login password:

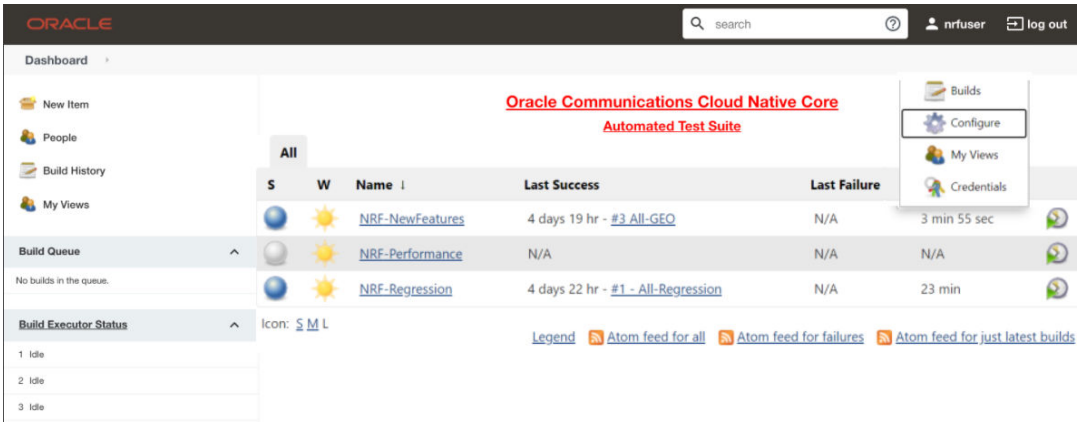
1. Log in to ATS application using the default login credentials. The home page of respective NF appears with its preconfigured pipelines as follows:

Figure 2-21 Sample Screen: NRF Home Page



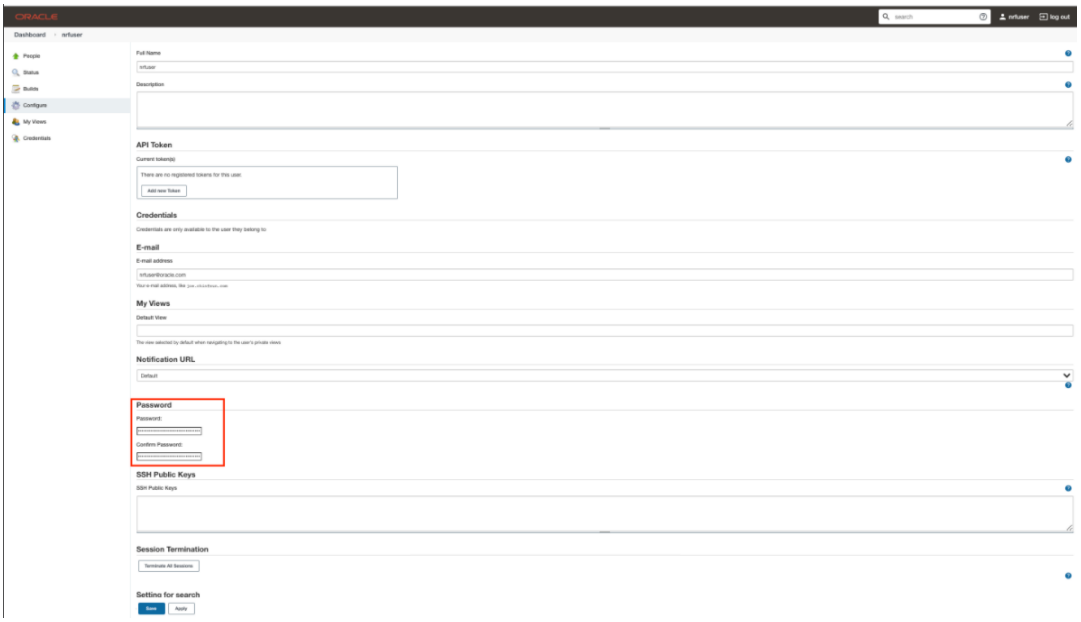
2. Hover over the user name and click the down arrow. Click **Configure** as follows:

Figure 2-22 Configure Option



3. The following page appears.

Figure 2-23 Logged-in User Detail



4. In the **Password** section, enter the new password in the **Password** and **Confirm Password** fields and click **Save**.

Thus, a new password is set for the user.

2.12 Parallel Test Execution

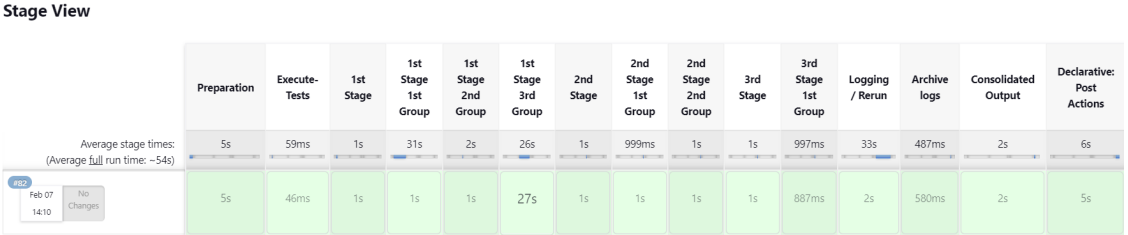
Parallel test execution enables you to perform multiple logically grouped tests simultaneously on the same System Under Test (SUT) to reduce the overall execution time of ATS.

ATS currently executes 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, then only the next stage will start the execution.

Pipeline Stage View

The pipeline stage view appears as follows:

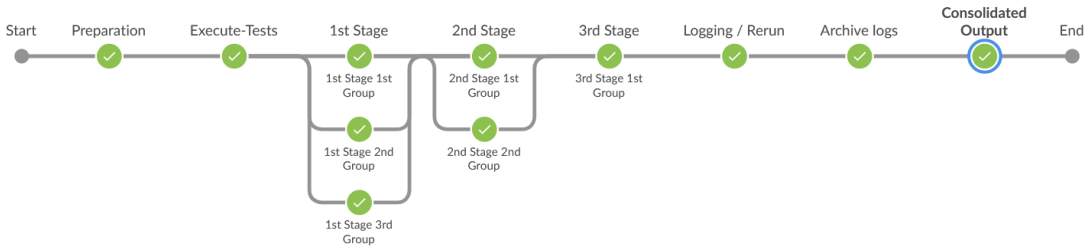
Figure 2-24 Pipeline Stage View



Pipeline Blue Ocean View

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

Figure 2-25 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 sections below for more details:

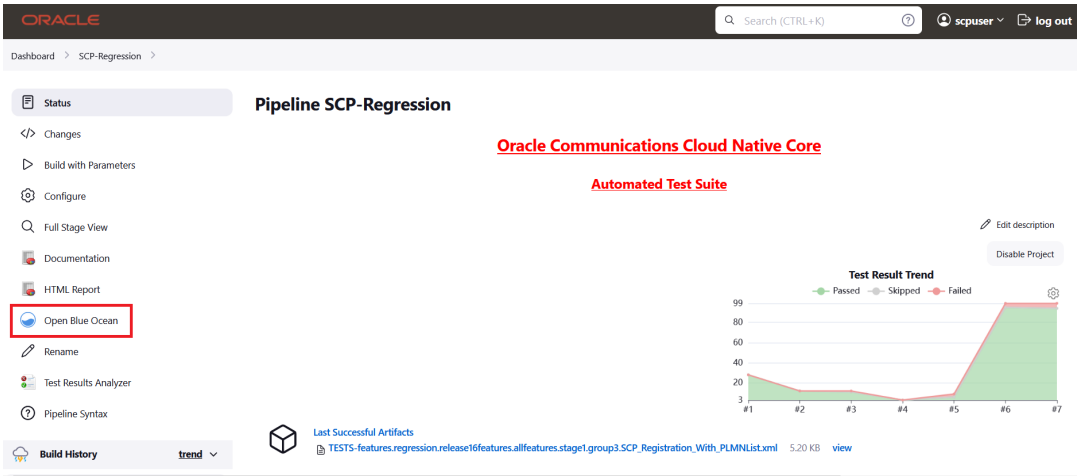
- Application Log Collection
- ATS API
- Final Summary Report, Build Color, and Application Log
- PCAP Log Collection

2.12.1 Downloading or Viewing Individual Group Logs

To download individual group logs:

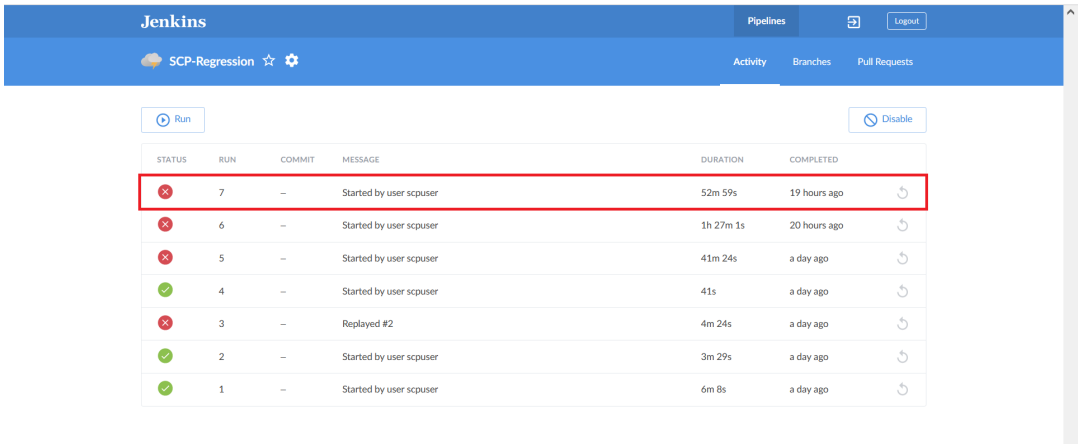
1. On the Jenkins pipeline page, click **Open Blue Ocean** in the left navigation pane.

Figure 2-26 Jenkins Pipeline Page



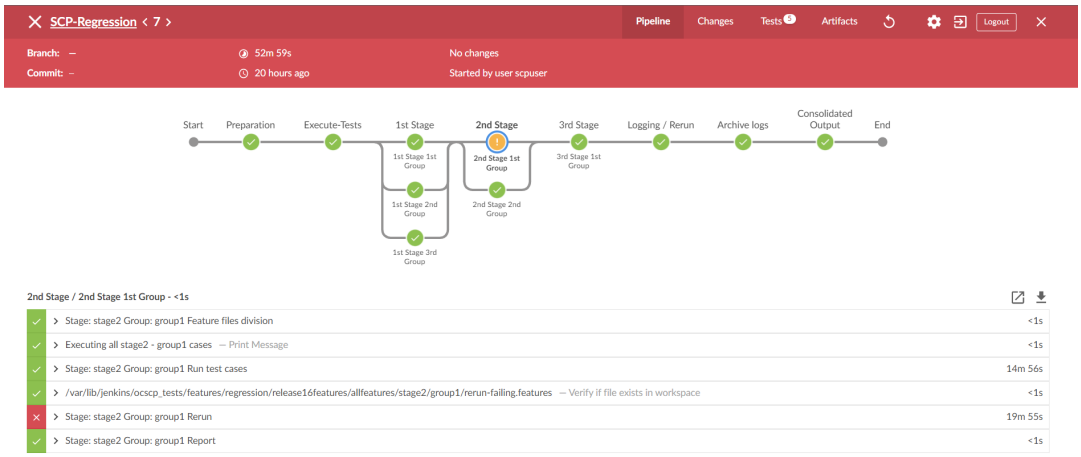
2. Click the desired build row on the Blue Ocean page.

Figure 2-27 Run the Build



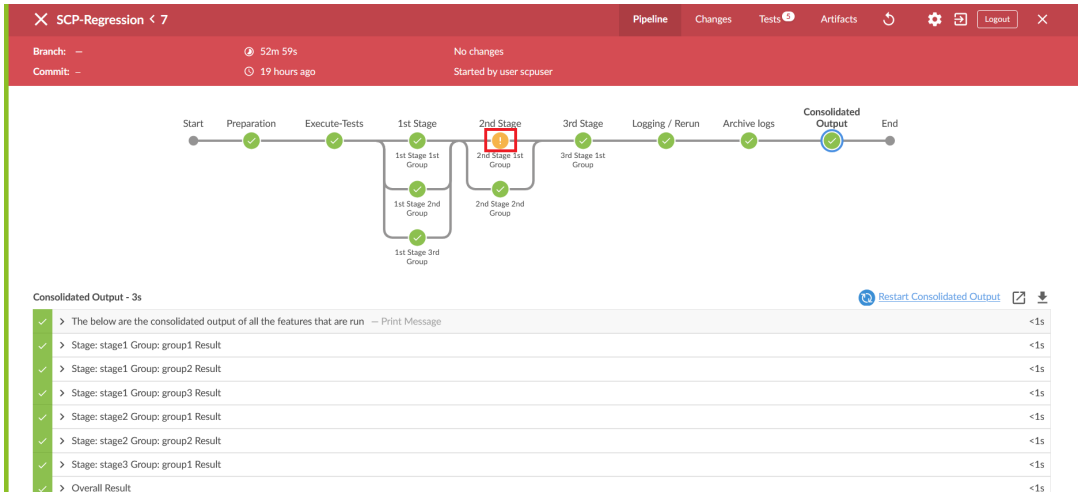
3. The selected build appears. The diagram displays the order in which the different stages, or groups, are executed.

Figure 2-28 Stage Execution



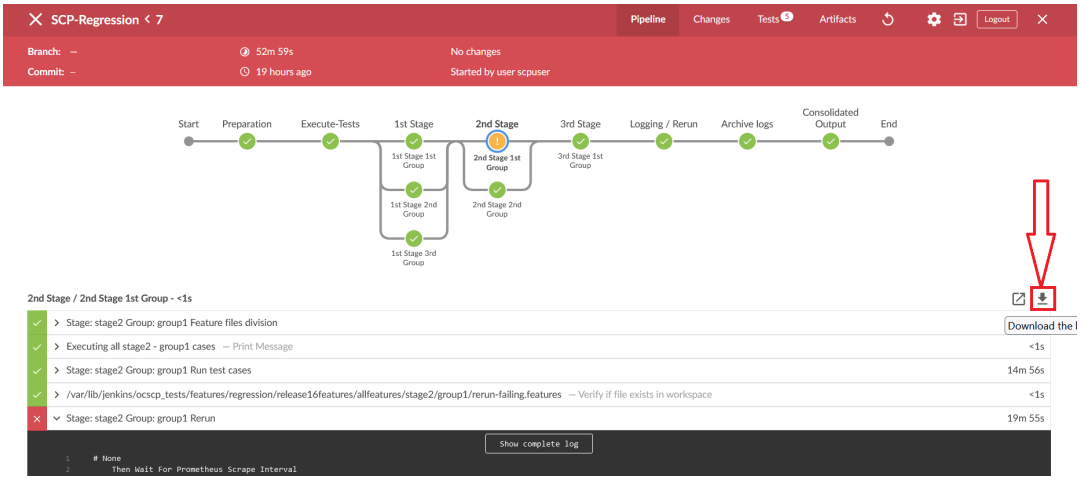
4. Click the desired group to download the logs.

Figure 2-29 Executed Groups



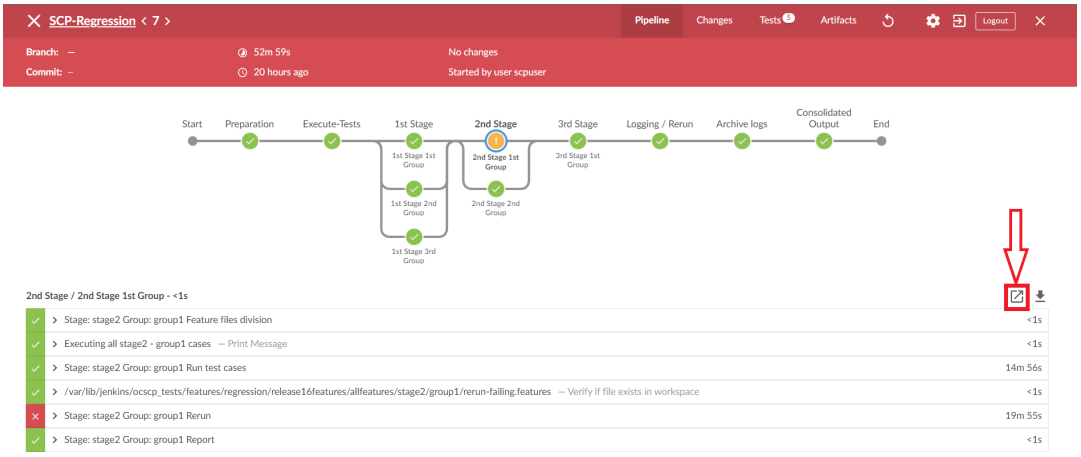
5. Click the **Download** icon on the bottom right of the pipeline. The log for the selected group is downloaded to the local system.

Figure 2-30 Download Logs



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

Figure 2-31 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.

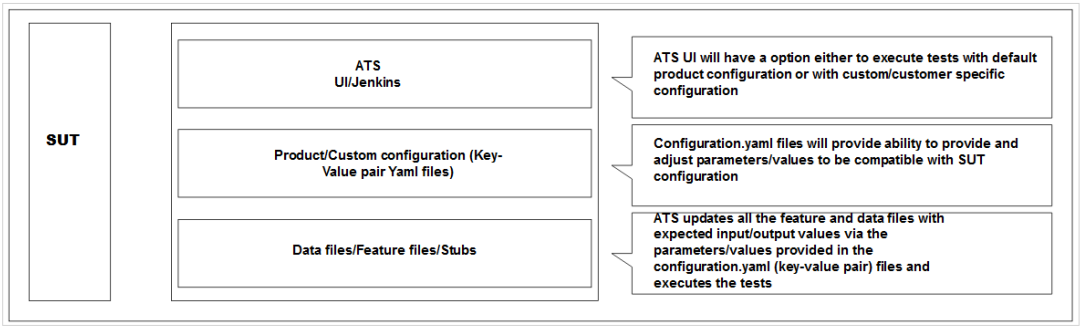
2.13 Parameterization

This feature enables users to provide or adjust values for the input and output parameters needed for the test cases to be compatible with SUT configuration. Users 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 it is 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. Users can maintain multiple versions or copies for the custom data folder to suite to varied or custom SUT configurations. With this feature ATS GUI has an option to either execute test cases with default product configuration or with custom configuration.

Key Functionality:

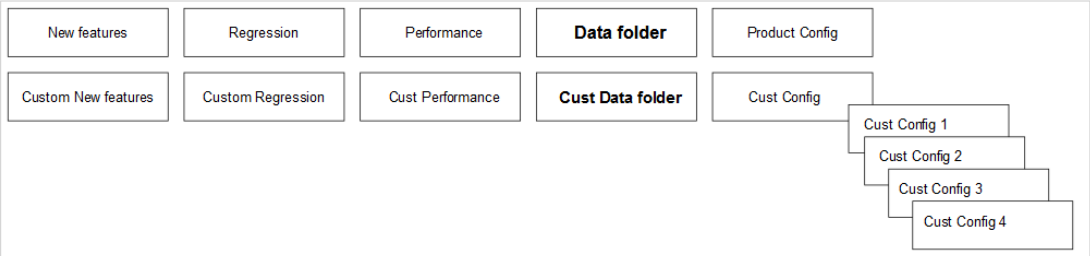
- Provides ability to define parameters and assign/adjust values to them so as to be compatible with SUT configuration.
- Provides ability to execute test cases either with default product configuration or custom configuration(s) - multiple custom configurations to match with varied SUT configurations.
- Simplified way to assign or adjust values (for input / output parameters) thru custom or default configuration yaml files (key-value pair files).
- Each feature file having its corresponding configuration file where-in the values for its input or output parameters can be defined or adjusted.
- Ability to create and maintain multiple configuration files to match multiple SUT configurations.

Figure 2-32 SUT Design Summary



The Parameterization feature enables users to provide or adjust values for the input and output parameters needed for the test cases to be compatible with SUT configuration. Users 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 it is compatible with SUT configuration. In addition to custom test case folders, this feature enables custom data, default product configuration, and custom configuration folders. Users can maintain multiple versions or copies of the custom data folder to suite to varied custom SUT configurations. With this feature ATS GUI has an option to either run test cases with default product configuration or with custom configuration.

Figure 2-33 Folder Structure



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 ATS package in every release
- The user can customize custom folders by:
  - Removing test cases not needed or as appropriate for users use
  - Adding new test cases as needed or as appropriate for users use
  - Removing or adding data files in the cust data folder or as appropriate for users use
  - Adjusting the parameters or values in the Key-value pair per yaml files in the custom config folder for test cases to run or pass with custom configured SUT
- The Product folders are always in-tact (unchanged) and the user can update the Custom folders

- The user can maintain multiple copies of Custom Configurations and can bring them to use needed or as appropriate for the SUT configuration

To Enable

For ATS to run the test cases with a particular custom configuration, rename or copy the Cust Config [1/2/3/N] folder to Cust Config folder. It always point to the Cust Config folder when selected to run test cases with custom configuration.

To Run ATS Test Cases

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

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

Figure 2-34 ATS Execution Flow

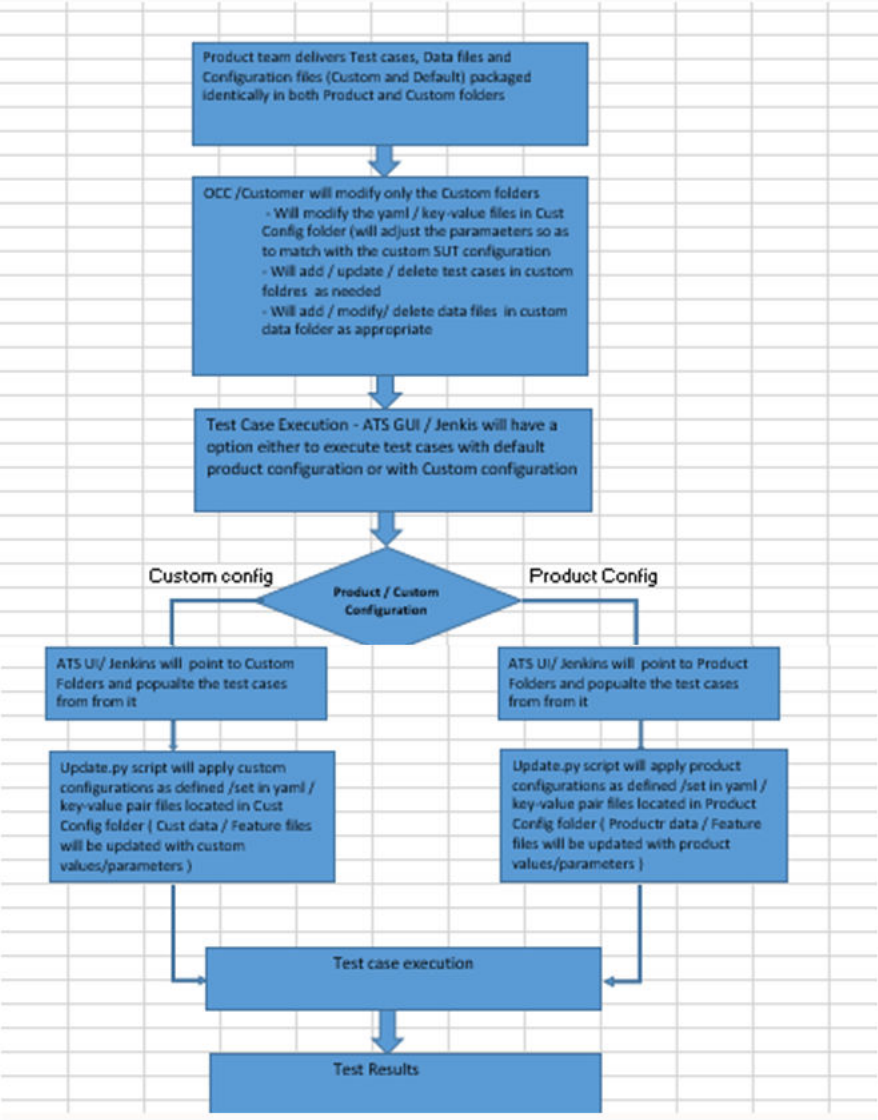
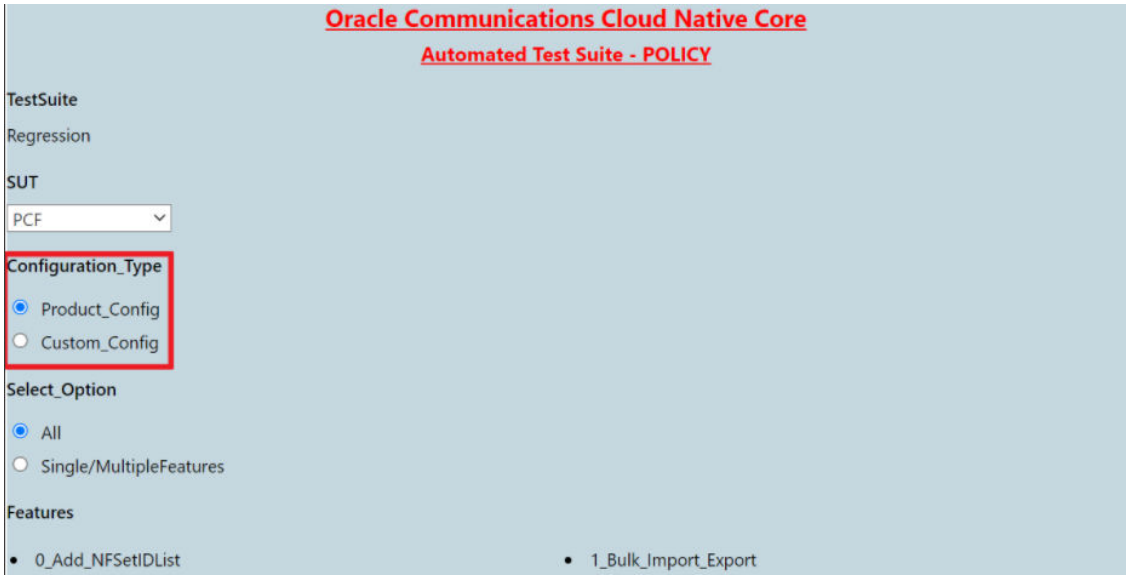


Figure 2-35 Sample: Configuration_Type



2.14 PCAP Log Collection

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

Figure 2-36 PCAP Logs Selection Option

1. The Debug tool should be enabled on SUT Pods while deploying the NF. The current name of the container is Tools.
2. Enable the pods/exec API group within the ATS service-account. Following are the ATS mandatory resource requirements:
   CPU: 3

   memory: 3GI
3. Once ATS is deployed, refer the `pre-TestConfig.sh` script to have the Jenkins variables.

4. Refer the framework related changes in the development branch `cleanup.py`.
5. Incorporate the `copyremotefile.java` and the `abortclean.py` scripts in the NF specific folder.
6. Have the `pcaplogs` folder cleanup in the archive log stage as below:

```
stage ('Archive logs') {
    steps {
         sh '''
         cd $JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/
         [ -d $JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/pcaplogs ]
&& zip -r pcaplogs.zip pcaplogs/
         rm -rf /var/lib/jenkins/.jenkins/jobs/$JOB_NAME/
builds/$BUILD_NUMBER/pcaplogs
            '''
    }
}
post {
        aborted {
                        script{
                                sh '/env/bin/python3 /var/lib/jenkins/<NF-
FOLDER>/abortclean.py'
                            }
                }
         always {
                         /* Existing Code */
                }
    }
```

7. Add the new Active choice Reference parameter for the pipeline jobs (single select).

```
return [
"YES:selected",
"NO"
]
```

Figure 2-37 Application Logs and PCAP Logs Selection

1. The Debug tool should be enabled on SUT Pods while deploying the NF. The current name of the container is Tools.
2. Enable the pods/exec API group within ATS service-account. Following are the ATS mandatory resource requirements:
   CPU: 3

   memory: 3GI
3. Once ATS is deployed, refer the `preTestConfig.sh` script to have the Jenkins variables.
4. Refer the framework related changes in the development branch `cleanup.py`.
5. Incorporate the `copyremotefile.java` and `abortclean.py` scripts in the NF specific folder.
6. Have the `pcaplogs applog` folder cleanup in the archive log stage as below:

```
stage ('Archive logs') {
    steps {
         sh '''
         cd $JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/
         [ -d $JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/pcaplogs ]
&& zip -r pcaplogs.zip pcaplogs/
         rm -rf /var/lib/jenkins/.jenkins/jobs/$JOB_NAME/
builds/$BUILD_NUMBER/pcaplogs
         [ -d $JENKINS_HOME/jobs/$JOB_NAME/builds/$BUILD_NUMBER/applog ]
&& zip -r applogs.zip applog/
         rm -rf /var/lib/jenkins/.jenkins/jobs/$JOB_NAME/
builds/$BUILD_NUMBER/applog
            '''
    }
}
post {
```

```
        aborted {
                        script{
                                sh '/env/bin/python3 /var/lib/jenkins/<NF-
FOLDER>/abortclean.py'
                               }
                }
         always {
                         /* Existing Code */
                }
    }
```

7. Add the three new Active choice Reference parameter for the pipeline jobs:
   a. Fetch_Log_Upon_Failure

      ```
      return [
      "YES:selected",
      "NO"
      ]
      ```

   b. Log_Type

      ```
      if(Fetch_Log_Upon_Failure.equals("NO"))
      {
      return ["Not Applicable:selected"]
      }
      else
      {
      return [
      "AppLog",
      "PcapLog [Debug Container Should be Running]"
      ]
      }
      ```

   c. Log_Level

      ```
      if(Fetch_Log_Upon_Failure.equals("NO"))
      {
      return ["Not Applicable:selected"]
      }
      else
      {
      return [
      "WARN:selected",
      "INFO",
      "DEBUG",
      "ERROR",
      "TRACE"
      ]
      }
      ```

2.15 Persistent Volume for 5G ATS

With the introduction of Persistent Volume, 5G ATS can retain historical build execution data, test cases, and ATS environment configurations.

ATS Packaging When Using Persistent Volume

- **Without Persistent Volume option:** ATS package includes ATS Image with test cases.
- **With Persistent Volume option:** ATS package includes 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 jobs data are packaged in the ATS image as well as tar file.

Process Flow

First Time Deployment

Initially when you deploy ATS (Example: PI-A ATS pod), you use PVC-A, which is provisioned and mounted to PI-A ATS pod. By default, the PVC-A is empty. So, you have to copy the data (ocslf_tests and jobs folder) 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 (Example: PI-B ATS pod), you use PVC-B, which is provisioned and mounted to PI-B ATS pod. By default, the PVC-B is empty and you have to copy the data (ocslf_tests and jobs folder) 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 can continue to retain the PVC-A. If you do not want backward porting, you can delete PVC-A.

Deploying Persistent Volume

Pre-installation Steps

1. Before deploying Persistent Volume, create a PVC in the same namespace where you have deployed ATS. You have to provide value 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 the subsequent releases. 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
persistentvolumeclaim/ocats-slf-1.9.0-pvc 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 2-38 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
```

In the above screenshot, verify that the **STATUS** is 'Bound' and rest of the parameters like NAME, CAPACITY, ACCESS MODES, STORAGECLASS etc are same as mentioned in the PVC creation command.

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

5. After creating persistent volume, change the following parameters in the values.yaml file (at ocats-udr location) to deploy persistent volume.

- Set the **PVEnabled** parameter to "true".
- Provide the value for **PVClaimName** parameter. The PVClaimName value should be same as used to create a PVC.

Post-Installation 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 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 folder. It can be different for different NFs.

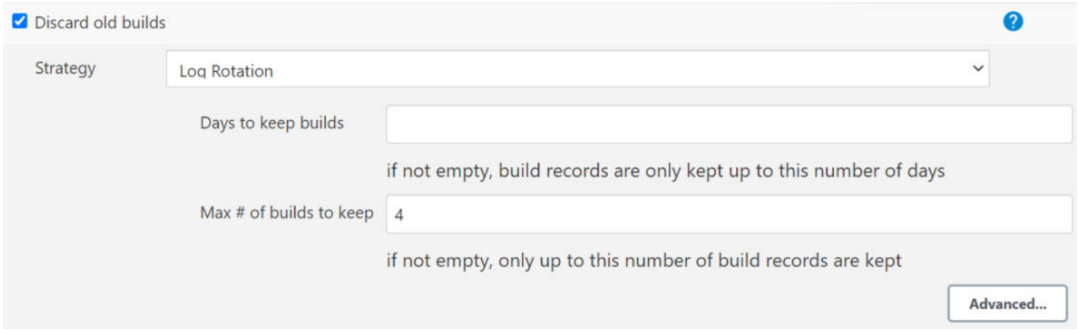
   b. Run the following set of commands to copy the required folders:
   `kubectl cp ats_data/jobs <namespace>/<pod-name>:/var/lib/jenkins/.jenkins/`

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

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

   Run the following command to restart the pods as one time activity.
   `kubectl delete po <pod-name> -n <namespace>`
2. Once 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 2-39 Discard Old Builds



> **Note**
>
> It is recommended to configure this option. If you do not enter any value for this option, then the application considers all the builds, which can be a huge number leading to complete consumption of Persistent Volume.

Backward Porting (deployment procedure for old release 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 purpose only and should not be considered as the subsequent release of POD deployment procedure.

The deployment procedure for old release PVC supported ATS pod is same, except that 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**

2.16 Test Results Analyzer

The Test Results Analyzer is a plugin available in ATS to view the 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:

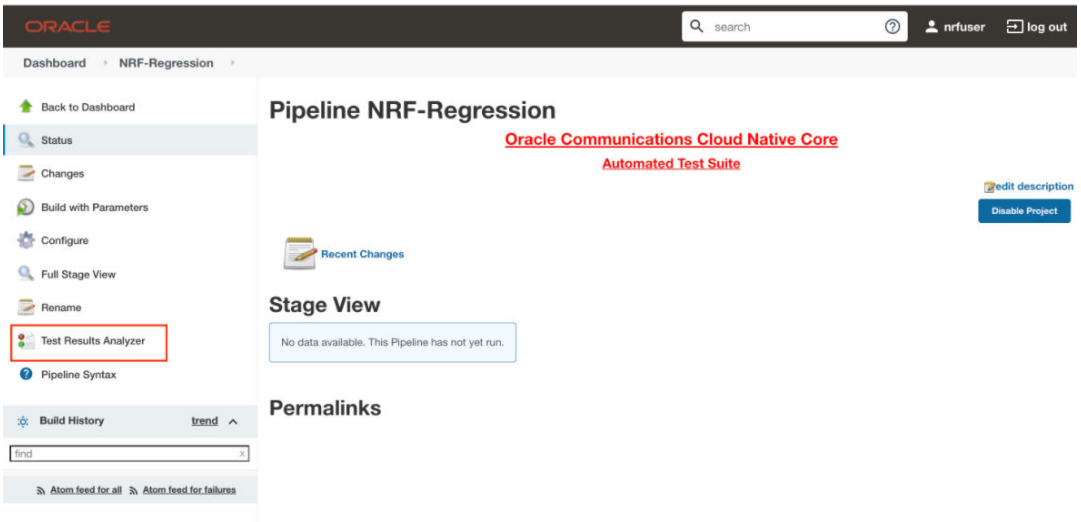
- **PASSED:** If the test case passes
- **FAILED:** If the test case fails
- **SKIPPED:** If the test case is skipped
- **N/A:** If the test cases is not executed in the current build

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 2-40 Test Results Analyzer Option



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

Figure 2-41 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 respective feature detail status report.

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

Figure 2-42 Test Result

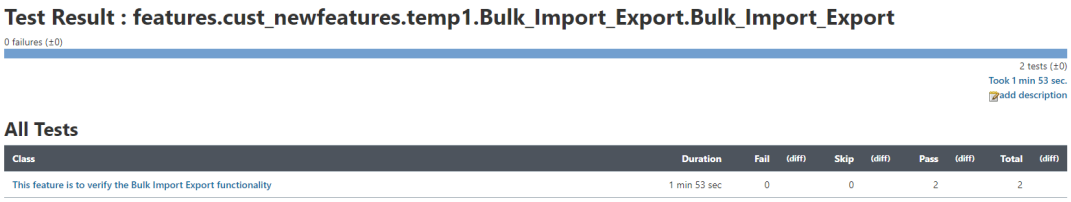
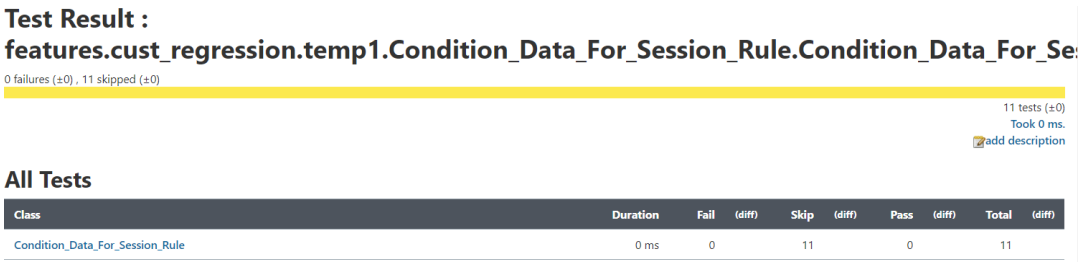


Figure 2-43 Test Result



4. In case of rerun, the test cases passed in main run but skipped in rerun are considered as 'Passed' in the Test Result Analyzer Report. The following screenshot depicts the Scenario, "Variant2_equal_smPolicySnssaiData,Variant2_exist_smPolicyData,Variant2_exist_smPolicyDnnData_dnn" where the test cases passed in main run but skipped in rerun are considered as 'PASSED' in general.

Figure 2-44 Test Results

Click 'Passed'. The following highlighted message means the test case is passed in the main run but skipped in rerun.

Figure 2-45 Test Result Info

2.17 Supports Test Case Mapping and Count

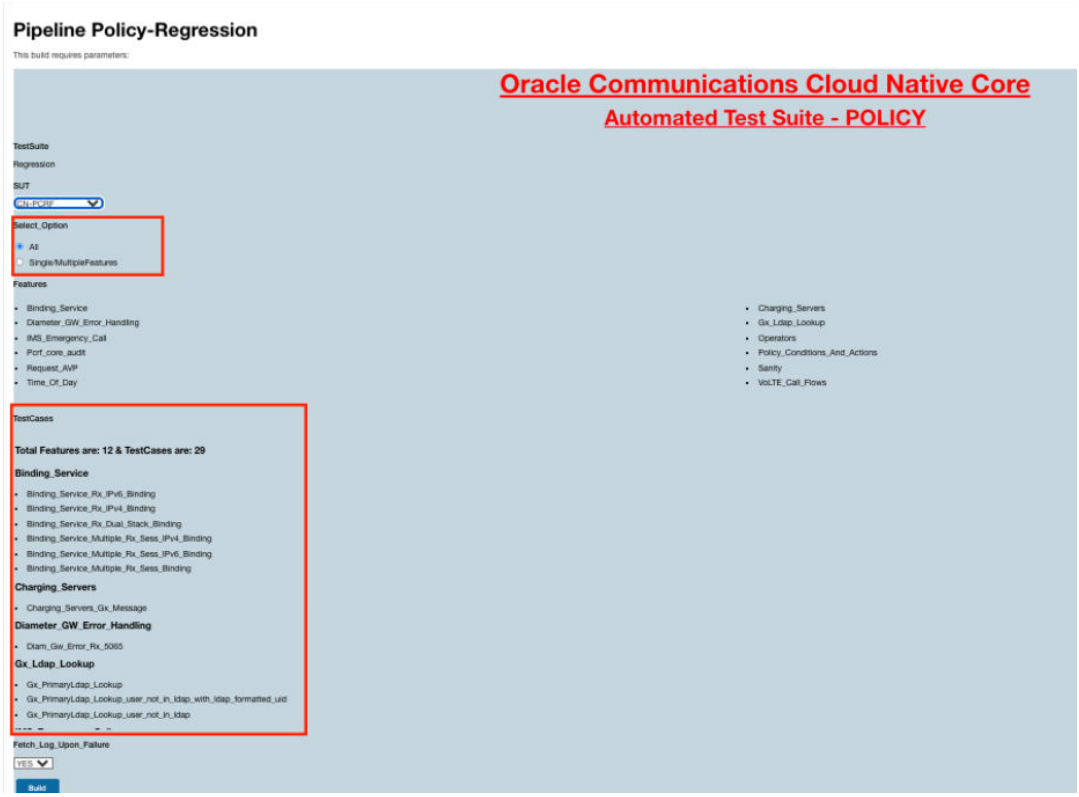
The 'Test Case Mapping and Count' feature displays total number of features, test cases or scenarios and its mapping to each feature in the ATS GUI.

Accessing Test Case Mapping and Count Feature

To access the 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**. The following image appears:

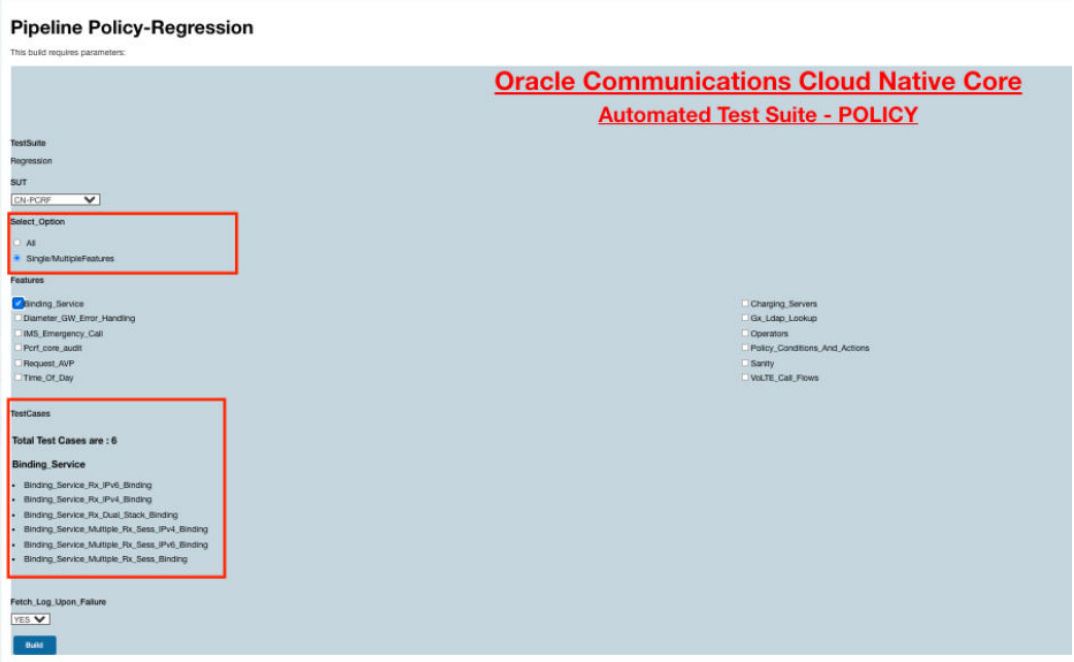
Figure 2-46 Test Case Mapping



In the above image, when **Select_Option** is selected as 'All', the **TestCases** details mapped to each feature appears.

If you select the **Select_Option** as 'Single/MultipleFeatures', the test cases details appear as follows:

Figure 2-47 Test Cases Details When Select_Option is Single/MultipleFeatures



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
Helm	3.1.2, 3.5.0, 3.6.3, 3.8.0
Podman	2.2.1, 3.2.3, 3.3.1
Oracle Communications Cloud Native Core, Cloud Native Environment (CNE)	Release 1.9.x /1.10.x/ 22.1.x, 23.1.0.0.0

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.

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 OCNWDAF 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	3	0	1	1	4	4	0
ocn-ats-nwdaf-notify-service	2	1	0	1	1	2	1	0

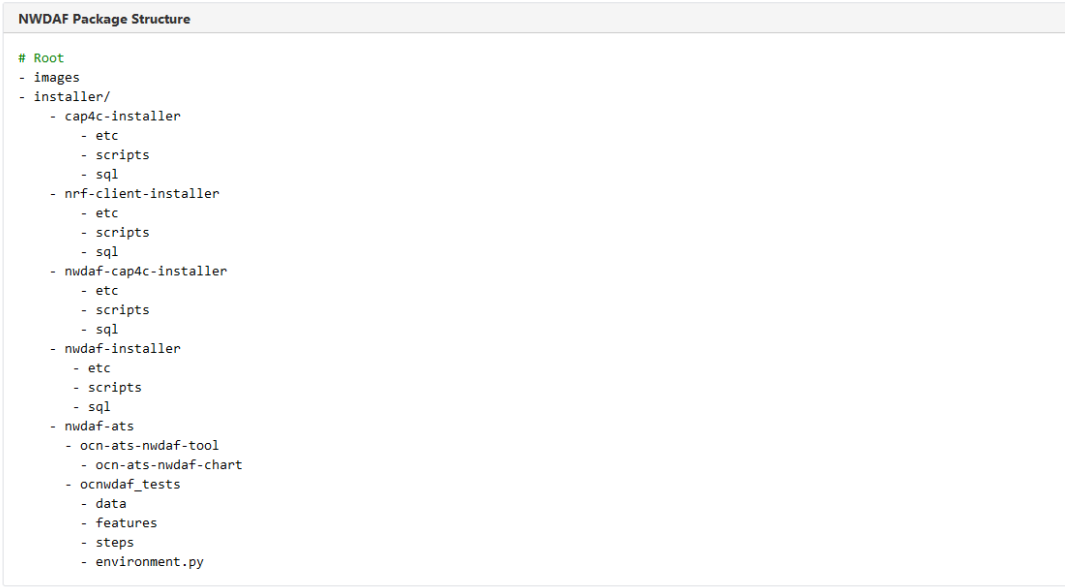
3.1.4 Downloading the ATS Package

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 - 5G* in the **Product** field.
5. Select *Oracle Communications Cloud Native Core Networks Data Analytics Function <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 NWDAF ATS package file.
10. Untar the zip file to access all the ATS Images. The **<p********_<release_number>_Tekelec>.zip**. The NWDAF directory has the following package structure:

Figure 3-1 OCNWDAF Package Structure



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

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:

Pre-requisites

- Oracle Linux 8 environment
- NWDAF 23.1.0.0.0 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 are pre-configured in the Helm charts. The following services are used for NWDAF ATS:

Table 3-3 NWDAF ATS Services

Service Name	Docker Image Name	Image Tag
`ocn-ats-nwdaf-notify-service`	`ocats-nwdaf-notify-api`	`23.1.0.0.0`
`ocn-ats-nwdaf-service`	`ocats-nwdaf-subs`	`23.1.0.0.0`

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
   ```

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

   - `ocats-nwdaf-notify-api`
   - `ocats-nwdaf-subs`

3. 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
   ```

4. 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.

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

   ```
   docker images
   ```

6. 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.

3.1.7 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-chart` directory. The namespace, docker image or tag can be updated in the `values.yaml` file.
2. Run the following command to deploy NWDAF ATS and its consumers:

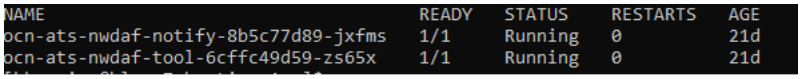
   ```
   helm install  --name<release_name> <chart name>.tgz --namespace
   <namespace_name> -f <values-yaml-file>
   ```

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

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

Example:

Figure 3-2 Sample Image



3.1.8 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>
```

3.1.9 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:

- Starts and ends with an alphanumeric character.
- Contains a maximum of "63" characters.
- Contains only alphanumeric characters or '-'.

3.2 Installing ATS for OCNADD

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

1. Downloading the ATS Package

2. Pushing the Images to Customer Docker Registry
3. Configuring ATS
4. Deploying ATS and Stub in Kubernetes Cluster
5. 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	37	49
ATS Totals	10	14
Grand Total OCNADD ATS	47	63

OCNADD Pods Resource Requirements Details

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

Table 3-5 OCNADD Pods Resource Requirements 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
ocnaddbackendrouter	1	1	1	1	1	1
ocnaddscpaggregation	3	2	2	1	3	2
ocnaddnrfaggregation	3	2	1	1	3	2
ocnaddseppaggregation	3	2	1	1	3	2
ocnaddadapter	3	4	8	1	3	4
ocnaddkafka	5	10	3	3	15	30
zookeeper	1	1	3	3	3	3
ocnaddgui	2	1	1	1	2	1
OCNADD SUT Totals					**37 CPU**	**49 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 Requirements details for OCNADD

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

Table 3-6 ATS Resource Requirements 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
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 Cloud Native Core - 5G* 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-23.1.0.tgz
ocats-ocnadd-tools-pkg-23.1.0-README.txt
ocats-ocnadd-tools-pkg-23.1.0.tgz.sha256
ocats-ocnadd-custom-configtemplates-23.1.0.zip
ocats-ocnadd-custom-configtemplates-23.1.0-README.txt
```

The `ocats-ocnadd-tools-pkg-23.1.0-README.txt` file has all the information required for the package.The `ocats-ocnadd-tools-pkg-23.1.0.tgz` file has the following images and charts packaged as tar files:

```
ocats-ocnadd-tools-pkg-23.1.0.tgz

|

|_ _ _ocats-ocnadd-pkg-23.1.0.tgz

|       |_ _ _ _ _ _ ocats-ocnadd-23.1.0.tgz (Helm Charts)

|       |_ _ _ _ _ _ ocats-ocnadd-image-23.1.0.tar (Docker Images)

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

|       |_ _ _ _ _ _ ocats-ocnadd-23.1.0.tgz.sha256

|       |_ _ _ _ _ _ ocats-ocnadd-image-23.1.0.tar.sha256

|       |_ _ _ _ _ _ ocats-ocnadd-data-23.1.0.tgz (ATS test scripts and
Jenkins data)

|       |_ _ _ _ _ _ ocats-ocnadd-data-23.1.0.tgz.sha256

|

|

|_ _ _ocstub-ocnadd-pkg-23.1.0.tgz

        |_ _ _ _ _ _ ocstub-ocnadd-23.1.0.tgz (Helm Charts)

        |_ _ _ _ _ _ ocstub-ocnadd-image-23.1.0.tar (Docker Images)

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

        |_ _ _ _ _ _ ocstub-ocnadd-23.1.0.tgz.sha256

        |_ _ _ _ _ _ ocstub-ocnadd-image-23.1.0.tar.sha256
```

In addition to the above images and charts, there is an `ocats-ocnadd-custom-configtemplates-23.1.0.tgz` file in the package file.

```
ocats-ocnadd-custom-configtemplates-23.1.0.tgz

      |

      |_ _ _ocats-ocnadd-custom-values_23.1.0.yaml (Custom values file for
installation)

      |

      |_ _ _ocats_ocnadd_custom_serviceaccount_23.1.0.yaml (Template to
create custom service account)
```

11. Copy the tar file to the OCCNE, 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-23.1.0.tgz
   ```
   The output of this command is:
   ```
   ocats-ocnadd-pkg-23.1.0.tgz
   ocstub-ocnadd-pkg-23.1.0.tgz
   ocats-ocnadd-custom-configtemplates-23.1.0.tgz
   ```
2. Run the following command to extract the helm charts and docker images of ATS.
   ```
   tar -xvf ocats-ocnadd-pkg-23.1.0.tgz
   ```
   The output of this command is:
   ```
   ocats-ocnadd-23.1.0.tgz
   ocats-ocnadd-23.1.0.tgz.sha256
   ocats-ocnadd-data-23.1.0.tgz
   ocats-ocnadd-data-23.1.0.tgz.sha256
   ocats-ocnadd-image-23.1.0.tar
   ocats-ocnadd-image-23.1.0.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-23.1.0.tgz
   ```
   The output of this command is:
   ```
   ocstub-ocnadd-image-23.1.0.tar
   ocstub-ocnadd-23.1.0.tgz.sha256
   ocstub-ocnadd-image-23.1.0.tar.sha256
   ocstub-ocnadd-23.1.0.tgz
   OCSTUB-ocnadd-Readme.txt
   OCSTUB_OCNADD_Installation_Readme.txt
   ```
4. Run the following command to extract the content of the custom configuration templates:
   ```
   tar -xvf ocats-ocnadd-custom-configtemplates-23.1.0.tgz
   ```
   The output of this command is:
   ```
   ocats-ocnadd-custom-values_23.1.0.yaml (Custom yaml file for deployment of
   OCATS-OCNADD)
   ```

```
ocats_ocnadd_custom_serviceaccount_23.1.0.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-23.1.0.tar', and push it to your registry.

```
$ docker load -i ocats-ocnadd-image-23.1.0.tar

$ docker tag docker.io/ocnaddats.repo/ocats-ocnadd:23.1.0 <local_registry>/
ocnaddats.repo/ocats-ocnadd:23.1.0

$ docker push <local_registry>/ocnaddats.repo/ocats-ocnadd:23.1.0
```

6. Run the following commands in your cluster to load the Stub docker images ocstub-ocnadd-image-23.1.0.tar and push it to your registry..

```
$ docker load -i ocstub-ocnadd-image-23.1.0.tar

$ docker tag docker.io/simulator.repo/ocddconsumer:1.0.20 <local_registry>/
simulator.repo/ocddconsumer:1.0.20
$ docker tag docker.io/simulator.repo/oraclenfproducer:1.0.14
<local_registry>/simulator.repo/oraclenfproducer:1.0.14
$ docker tag docker.io/utils.repo/jdk17-openssl:1.0.6 <local_registry>/
utils.repo/jdk17-openssl:1.0.6

$ docker push <local_registry>/simulator.repo/ocddconsumer:1.0.20
$ docker push <local_registry>/simulator.repo/oraclenfproducer:1.0.14
$ docker push <local_registry>/utils.repo/jdk17-openssl:1.0.6
```

7. 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"
```

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

3.2.5 Deploying ATS and Stub in Kubernetes Cluster

> **Note**
>
> - It is important to ensure that all the three components; ATS, Stub and OCNADD are in the same namespace.
> - For the ATS use case, the minimum replica set of OCNADD Adapter deployment must be updated to 1

ATS and Stub support Helm deployment.

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:

```
helm install -name <release_name> ocats-ocnadd-23.1.0.tgz --namespace
<namespace_name> -f <values-yaml-file>
```

Example:

```
helm install -name ocats ocats-ocnadd-23.1.0.tgz --namespace ocnadd -f ocats-
ocnadd-custom-values.yaml
```

Deploying Stubs:

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

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

Example:

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

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
```

To check service deployment:

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

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 custom `values.yaml` file as per the configuration to be tested. The custom `values.yaml` file is available with the NWDAF documentation package. Ensure that you deploy the NWDAF for ATS using the correct custom value file based on the selected NWDAF configuration.

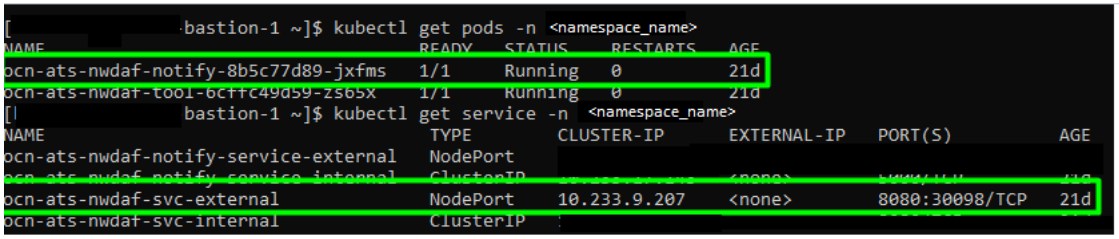
Logging into ATS

Running ATS

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

For example:

Figure 4-1 Verify ATS Deployment

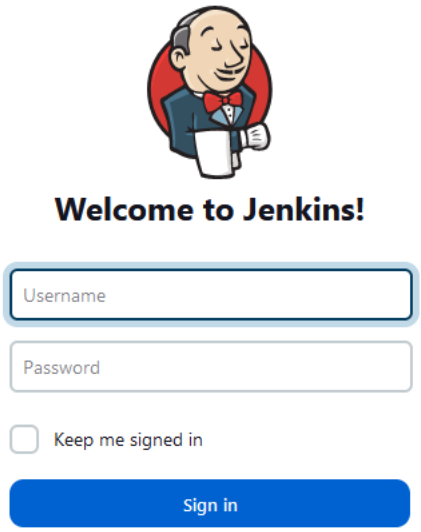


For more information on verifying ATS deployment, see Verifying ATS Deployment .

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

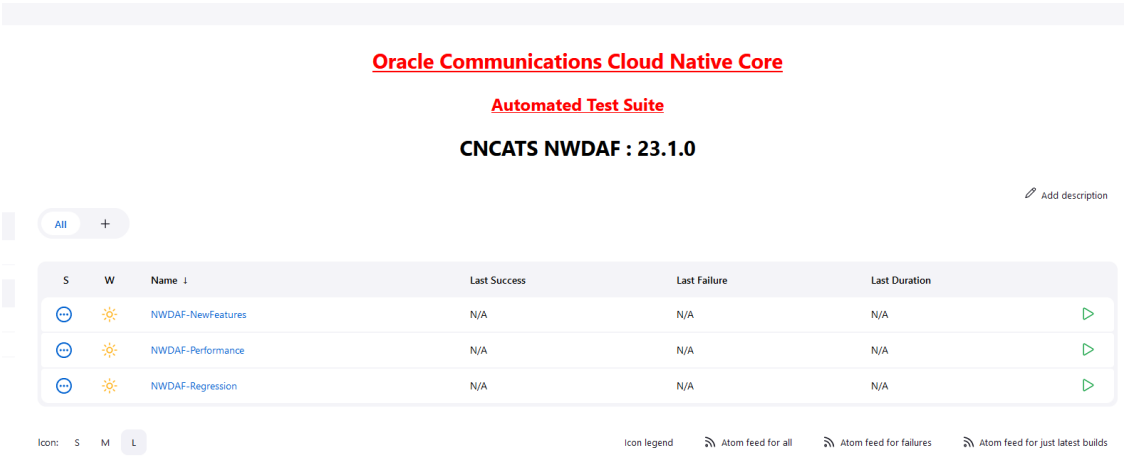
To log in to ATS, open a browser and provide the IP Address and port details as <Worker-Node-IP>:<Node-Port-of-ATS>.

Figure 4-2 ATS Login



To run ATS, enter the login credentials. Click **Sign in**. A screen displaying the NWDAF pre-configured pipelines appears.

Figure 4-3 Pipelines



The NWDAF ATS has three configured pipelines:

- NWDAF-NewFeatures

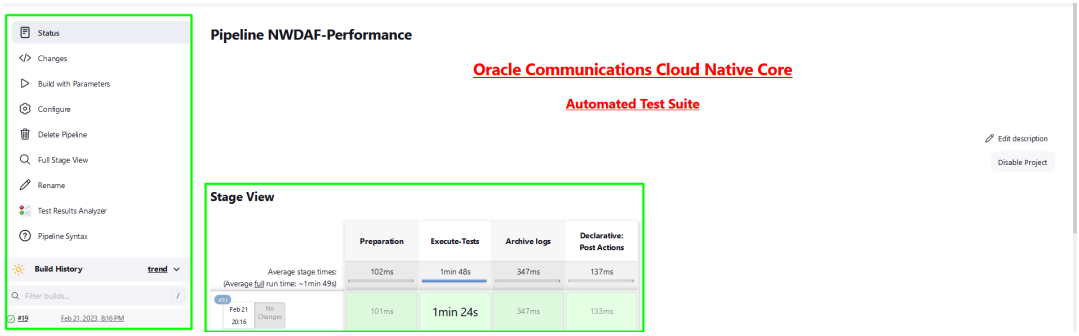
- NWDAF-Performance
- NWDAF-Regression

Configure NWDAF Pipelines

To run configure a pipeline perform the following steps:

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

Figure 4-4 Pipeline Configuration



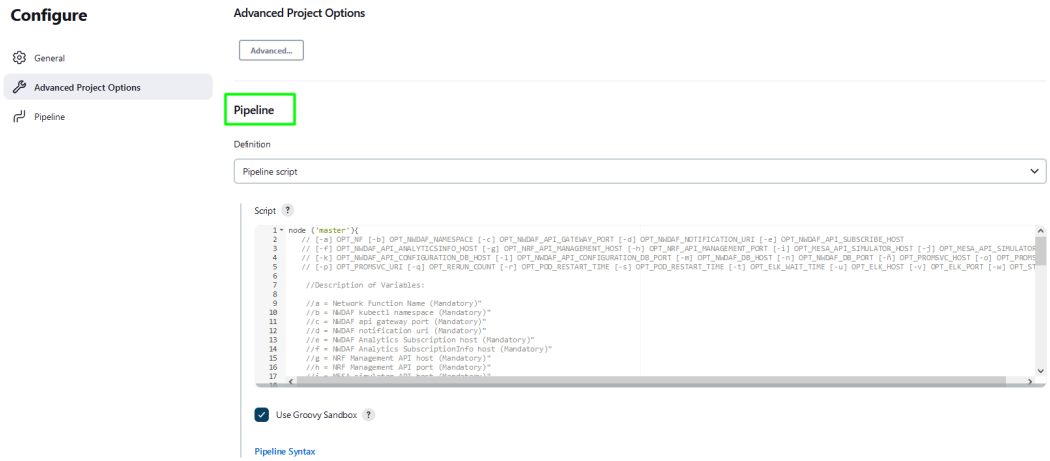
In the above screen:

- 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 **Stage View** represents the previously executed 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.

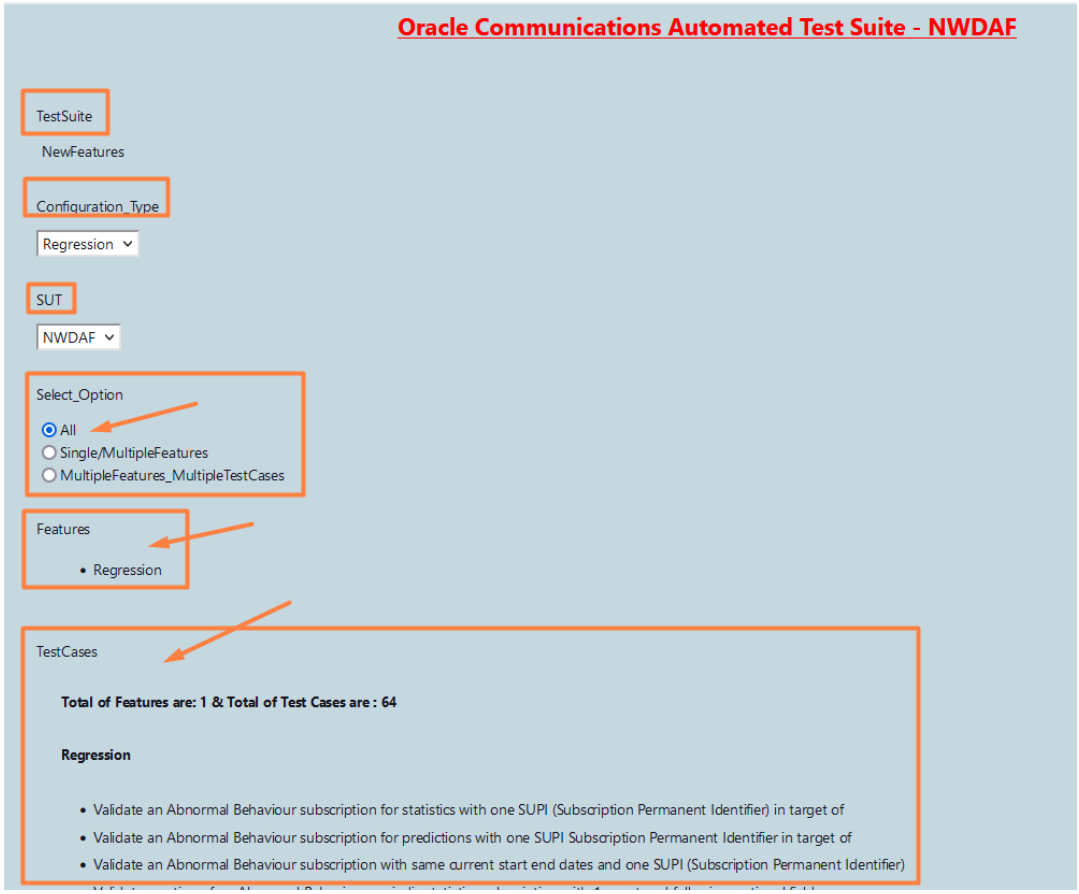
Figure 4-5 Configure Pipeline



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

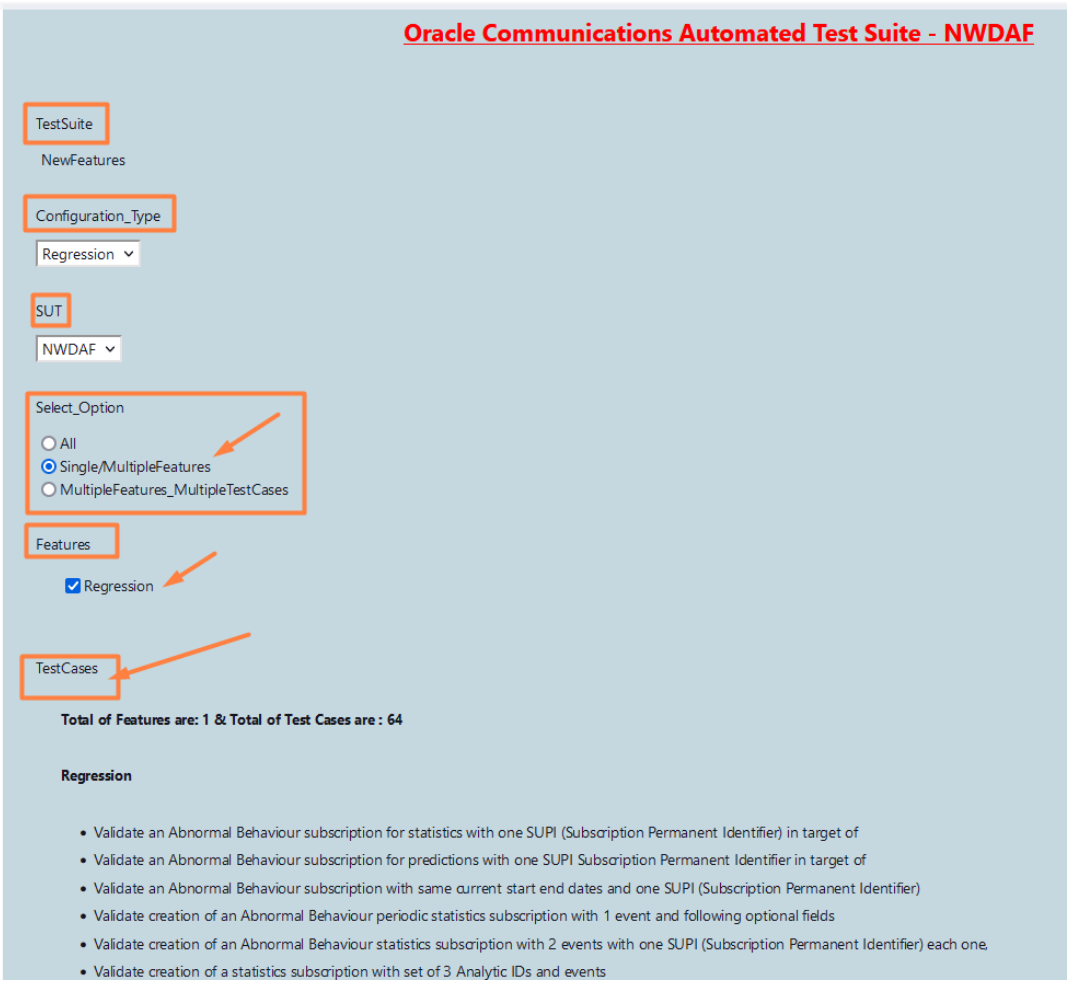
3. Depending on the pipeline you want to configure, use the respective script. The following scripts are available at: `/var/lib/jenkins/ocnwdaf_tests/pipeline_scripts`:
   - NewFeatures_Pipeline_script.txt
   - Regression_Pipeline_script.txt
   - Performance_Pipeline_script.txt
4. Update the script of the pipeline you want to configure, and update the parameters according to the ATS and NWDAF environment you want to test.
5. Click **Save**.
6. To run the test cases, click **Build with Parameters**.
7. The TestSuite multi-option page is displayed, ensure the SUT value is NWDAF.

Figure 4-6 TestSuite Page



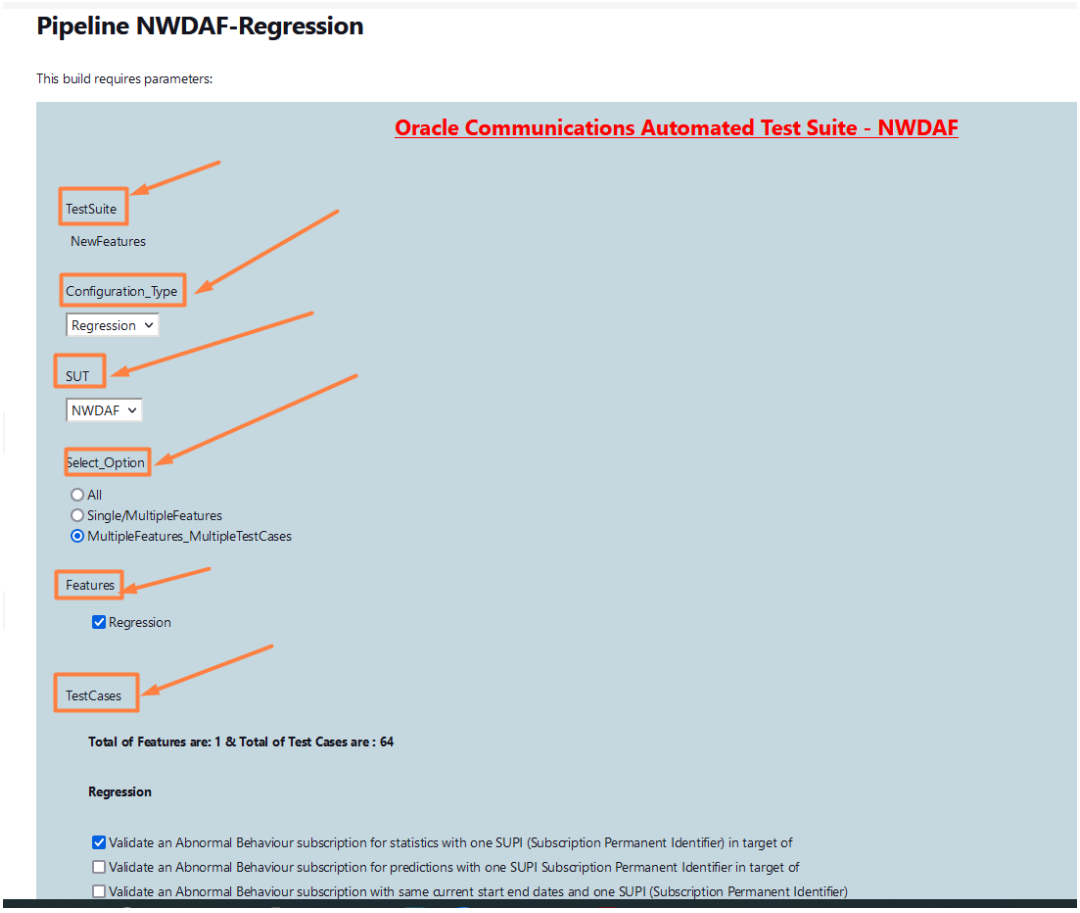
8. To run all scripts available, select **All** and click **Build**.
9. To run scripts on a specific feature, select **Single/MultipleFeatures** option and select the **Feature** you want to run the test on and click **Build**.

Figure 4-7 Single/Multiple Features



10. 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 4-8 Multiple Features and Testcases



11. The **Build History** menu displays the list of running jobs, select the latest job displayed in the list.
12. Click the **Console Output** option for this job. The jobs progress can be visualized in the log.

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 into ATS
- OCNADD-NewFeatures Pipeline

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 OCNADD, ATS, and Stub must be deployed in the same namespace.

4.2.2 Logging into ATS

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

```
[ocnadd@k8s-bastion ~]$ helm status ocats -n ocnadd
NAME: ocats
LAST DEPLOYED: Sat Nov  3 03:48:27 2022
NAMESPACE: ocnadd
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 mad | grep ocats
ocats-ocats-ocnadd-54ffddb548-4j8cx         1/1     Running     0          9h

[ocnadd@k8s-bastion ~]$ kubectl get svc -n mad | 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>.

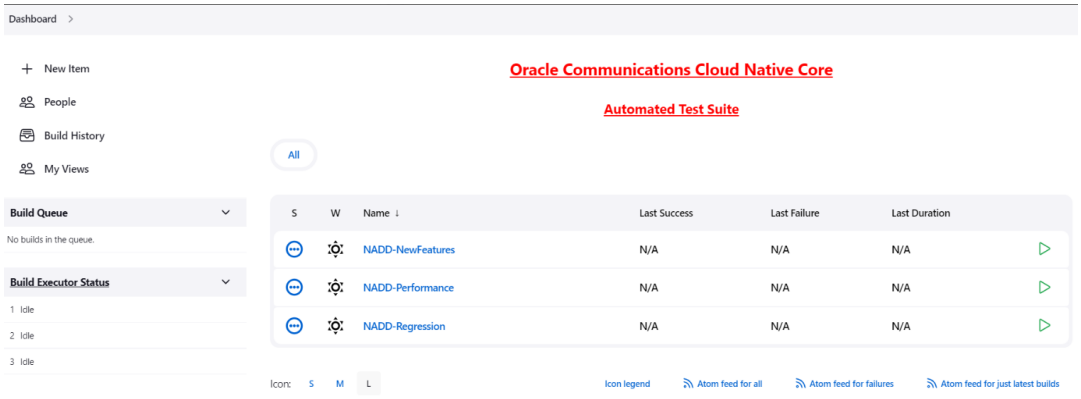
Figure 4-9 ATS Login

Running ATS

To run ATS:

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

Figure 4-10 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 - 23.1.0.
- **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.

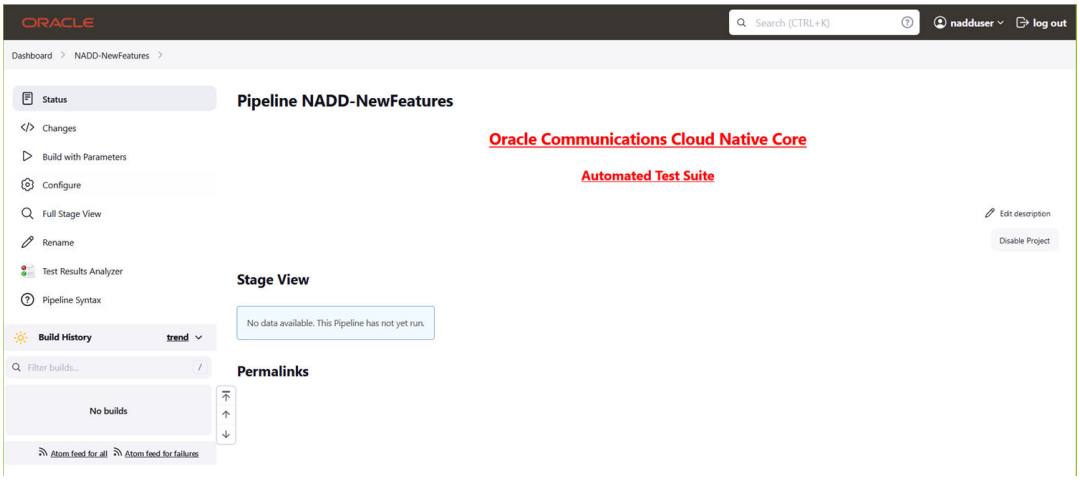
4.2.3 OCNADD-NewFeatures Pipeline

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 4-11 Configuring OCNADD-New Features



In the above screen:

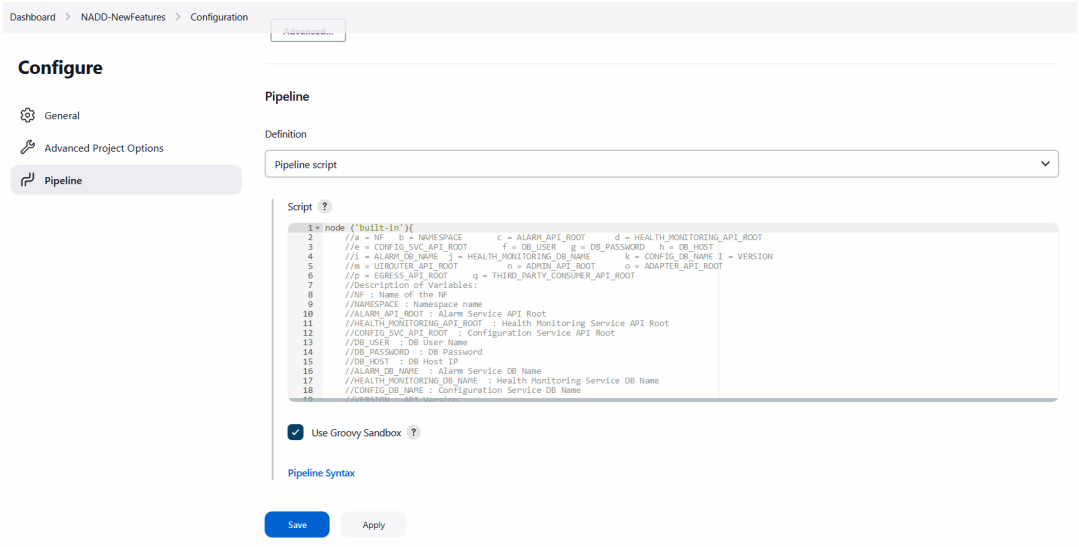
- Click **Configure** to configure OCNADD-New Features.
- Click **Build History** box to view all the previous pipeline executions, and the Console Output of each execution.
- The **Stage View** represents the previously executed pipelines for reference.
- The **Test Results Analyzer** is the plugin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

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 4-12 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 = VERSION    d = DB_HOST   e =
DB_USER    f = DB_PASSWORD
            //g= ALARM_DB_NAME    h = HEALTH_MONITORING_DB_NAME    i =
CONFIG_DB_NAME
            //j= ALARM_API_ROOT    k = HEALTH_MONITORING_API_ROOT    l =
CONFIG_SVC_API_ROOT
            //m= UIROUTER_API_ROOT    n = ADMIN_API_ROOT    o =
THIRD_PARTY_CONSUMER_API_ROOT
            //p= BACKUP_RESTORE_IMG_PATH    q = ALERT_MANAGER_URI
            //Descriptionof Variables:
            //NF: Name of the NF
            //NAMESPACE: Namespace name
            //VERSION: API Version
            //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 forBackup restore
image
            //ALERT_MANAGER_URI: Alert Manager API Root

           sh '''
                sh /var/lib/jenkins/ocnadd_tests/preTestConfig-NewFeatures-
NADD.sh \
                -a NADD \
                -b v2 \
                -c <ocnadd-namespace> \
                -d <DB_HOST> \
                -e <DB_USER> \
                -f <DB_PASSWORD> \
                -g alarm_schema \
                -h healthdb_schema \
                -i configuration_schema\
                -j ocnaddalarm:9099 \
                -k ocnaddhealthmonitoring:12591 \
                -l ocnaddconfiguration:12590 \
                -m ocnaddbackendrouter:8988\
                -n ocnaddadminservice:9181 \
                -o ocnaddthirdpartyconsumer:9094 \
                -p <repo-path>/ocdd.repo/ocnaddbackuprestore:<tag>\
                -q occne-kube-prom-stack-kube-alertmanager.occne-
infra.svc.<domainName>:80/<clusterName>\
            '''
            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).(Must not be modified)
- b: Namespace in which the NADD is deployed. (ocnadd)
- c: API Version. (v2)
- d: DB Host IP as provided during deployment of NADD. (10.XX.XX.XX)
- e: DB Username as provided during deployment of NADD.
- f: DB password as provided during deployment of NADD.
- g: DB Schema Name of ocnaddalarm microservice as provided during deployment of NADD. (alarm_schema)
- h: DB Schema Name of ocnaddhealthmonitoring microservice as provided during deployment of NADD. (healthdb_schema)
- i: DB Schema Name of ocnaddconfiguration microservice as provided during deployment of NADD. (configuration_schema)
- j: API root endpoint to reach ocnaddalarm microservice of NADD. (<Worker-Node-IP>:<Node-Port-of-ocnaddalarm>) or default value
- k: API root endpoint to reach ocnaddhealthmonitoring microservice of NADD. (<Worker-Node-IP>:<Node-Port-of-ocnaddhealthmonitoring>) or default value

- l: API root endpoint to reach ocnaddconfiguration microservice of NADD. (<Worker-Node-IP>:<Node-Port-of-ocnaddconfiguration>) or default value
- m: API root endpoint to reach ocnaddbackendrouter microservice of NADD. (Not used in the current release, use the default value)
- n: API root endpoint to reach ocnaddadminservice microservice of NADD. (Not used in the current release, use the default value)
- o: API root endpoint to reach ocnaddthirdpartyconsumer microservice of NADD. (<Worker-Node-IP>:<Node-Port-of-ocnaddthirdpartyconsumer>) or default value
- p: Repository path for ocnaddbackuprestore image.
- q: API root endpoint to reach alert manager microservice.

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 4-13 Pipeline NADD_NewFeatures

2. Select **Configuration_Type** as **Product_Config**.
3. In **Select_Option**:
   - Select **All** to run all the feature test cases and click the **Build** button to execute the pipeline
   - Choose **Single/MultipleFeatures** to run the specific feature test cases and click the **Build** button to execute the pipeline