Oracle® Communications 5G Automated Testing Suite Guide

Release 1.3.2
F36579-01
October 2020

Oracle Communications 5G Automated Testing Suite Guide , Release 1.3.2

F36579-01

Contents

What's New in This Guide

This section shares the list of new features introduced in every ATS release. For more release specific information, you can refer to release notes.

Release 1.3.2

Following new NF and features are added to ATS 1.3.2 release:

Table Features

Feature	BSF
Service Mesh Support	Yes
RBAC authorization	Role Binding
Custom Folder Implementation	Yes
New TestCases added to Jenkins Pipeline	Yes
Number of new TestCases added to Jenkins Pipeline	Provides a total of **12 scenarios** clubbed together in **3 feature files** of BSF ATS - 1.6.0 New Feature pipeline.
Previous Release TestCases	Not Applicable

Release 1.3.1

Following new NF and features are added to ATS 1.3.1 release:

Table Features

Feature	SEPP	Policy
Service Mesh Support	Not Applicable	Yes
RBAC authorization	Role Binding	Same as Previous Release (Role Binding)
Custom Folder Implementation	Yes	Yes
New TestCases added to Jenkins Pipeline	Yes	Yes
Number of new TestCases added to Jenkins Pipeline	Provides a total of **24 scenarios** clubbed together in **23 feature files** of SEPP ATS - 1.4 New Feature pipeline.	Provides a total of **44 scenarios** clubbed together in **10 feature files** of Policy ATS - 1.8.1 New Feature pipeline.
Previous Release TestCases	Not Applicable	Provides total of **95 scenarios** clubbed together in **38 feature files** of Policy-Regression pipeline.

Release 1.3.0

Following new features are added to ATS 1.3.0 release:

Table Features

Feature	NRF	NSSF	Policy	SCP	UDR
Service Mesh Support	Yes	Not Applicable	Yes	Yes	Yes
RBAC authorization	Same as Previous Release (Role Binding)	Same as Previous Release (Cluster Role Binding)	Same as Previous Release (Role Binding)	Same as Previous Release (Role Binding)	Same as Previous Release (Role Binding)
Custom Folder Implementation	Yes	Not Applicable	Yes	Yes	Yes
New TestCases added to Jenkins Pipeline	Yes	Not Applicable	Yes	Yes	Yes
Number of new TestCases added to Jenkins Pipeline	Provides a total of **55 scenarios** clubbed together in **21 feature files** of NRF ATS - 1.8.0 New Feature pipeline.	Not Applicable	Provides a total of **33 scenarios** clubbed together in **7 feature files** of Policy ATS - 1.8.0 New Feature pipeline.	Provides a total of **75 scenarios** clubbed together in **3 feature files** of SCP ATS - 1.8.0 New Feature pipeline.	Provides a total of **39 scenarios** clubbed together in **5 feature files** of ProvGw ATS - 1.8.0 New Feature pipeline.
Previous Release TestCases	Provides a total of **572 scenarios** clubbed together in **243 feature files** of NRF Regression pipeline.	Not Applicable	Provides total of **95 scenarios** clubbed together in **38 feature files** of Policy-Regression pipeline.	Provides **25 scenarios** in **4 feature files**.	Provides **45 scenarios** in **4 feature files** of UDR-Regression pipeline.

1

Understanding Automated Testing Suite (ATS)

In this chapter, you will get an overview about ATS, its need and its features.

Automated Testing Suite Overview

Automated Testing Suite (ATS) allows you to execute software test cases using an automated testing tool and then, compares the actual results with the expected or predicted results. In this process, there is no intervention from the user.

ATS for 5G Network Functions

For 5G Network Functions (NFs), ATS is built using **Oracle Linux 7-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, user are provided with the ATS image, simulator images, and test cases for the specific NF. All these are handed over to the customer as a fully automated suite so that they can directly perform Lab deployment and testing. You can combine it with any other **Continuous Integration (CI) pipeline** with minimal changes. Since, 5G ATS uses Jenkins as GUI.

Why Automated Testing Suite in 5G NFs?

Through Automated Testing Suite (ATS), Oracle Communications aims at providing an end-to-end solution to its customers for deploying and testing its 5G-NFs.

This guide covers implementation of ATS for the following 5G NFs:

- Network Repository Function (NRF)
- Network Slice Selection Function (NSSF)
- Policy
- Service Communication Proxy (SCP)
- Security Edge Protection Proxy (SEPP)
- Unified Data Repository (UDR)

ATS Features

The ATS features are as follows:

- Provides an end-to-end solution to the customers for testing Oracle Communications 5G-NFs. The ATS package includes:
  - Test scripts and docker images of test container.

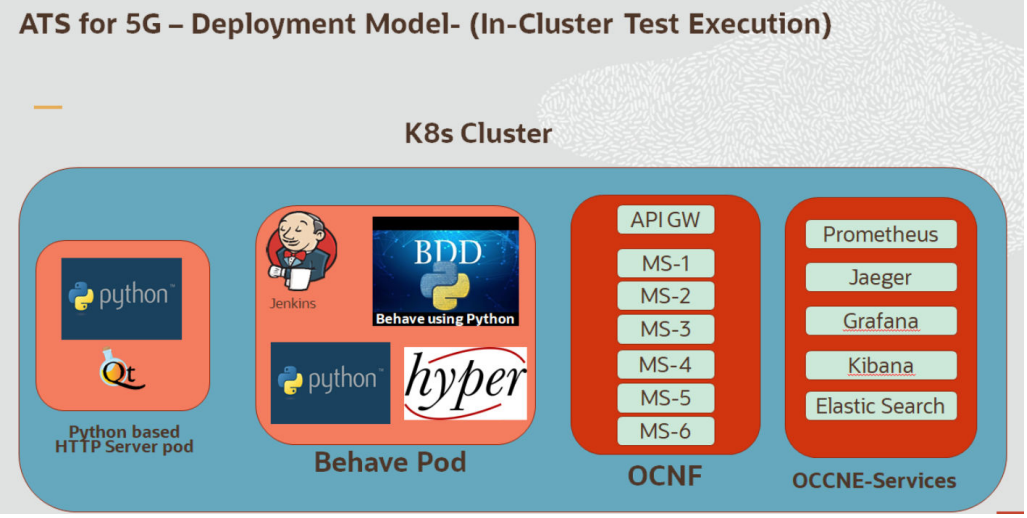
    * The docker images have complete framework and libraries installed, which is common for all NFs working with BDD framework.
  - Docker image of HTTP Server simulator
  - Helm chart to deploy the ATS (delivered as a tar file)
  - Readme text file (.txt file)
- Enables all the NF teams with the basic environment, framework and a GUI (Jenkins) to execute all the functional test cases.

Deployment Model (In-Cluster)

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

> **Note:**
>
> The ATS 1.3 package supports in-cluster deployment.

Figure 1-1 In-Cluster Deployment Model



> **Note:**
>
> GO Language is used to create stubs for Policy ATS and SCP ATS.

2
NF ATS Installation Procedure

The following sections describe how to install ATS for the following network functions:

- BSF
- NRF
- NSSF
- Policy
- SCP
- SEPP
- UDR

BSF ATS Installation Procedure

The BSF ATS installation procedure covers two steps:

1. Locating and downloading the ATS images.
2. Deploying ATS images.

This includes installation of stub, ATS in BSF's namespace (ocbsf).

Downloading ATS Image

To download the ATS Image from MOS:

1. Login to My Oracle Support with your credentials.
2. Select **Patches and Updates** tab to locate the patch.
3. In **Patch Search** window, click **Product or Family (Advanced)**.
4. Enter *Oracle Communications Cloud Native Core - 5G* in **Product** field, select *Oracle Communications Cloud Native Binding Support Function 1.6.0.0.0* from **Release** drop-down.
5. Click on **Search**. The **Patch Advanced Search Results** displays a list of releases.
6. Select the required patch from the search results. The Patch Details window opens.
7. Click **Download**. File Download window appears.
8. Click the **<p********_<release_number>_Tekelec>.zip** file to downlaod the BSF ATS package file.
9. Untar the zip file to access all the ATS Images.

10. The `ocats-policy-tools-1.6.0.0.0.tgz` file has following images and charts packaged as tar files:

```
ocats-policy-tools-1.6.0.0.0.tgz

|

|_ _ _ocats-policy-pkg-1.6.0.0.0.tgz

|      |_ _ _ _ _ _ ocats-policy-1.6.0.tgz (Helm Charts)

|      |_ _ _ _ _ _ ocats-policy-images-1.6.0.tar (Docker Images)

_ _ _ocamf-pkg-1.1.0.0.0.tgz

|      |_ _ _ _ _ _ ocamf-stub-1.1.0.tgz (Helm Charts)

|      |_ _ _ _ _ _ ocamf-stub-image-1.1.0.tar (Docker Images)|

|_ _ _ocstub-pkg-1.1.0.0.0.tgz

|      |_ _ _ _ _ _ ocstub-go-1.1.0.tgz(Helm Charts)

|      |_ _ _ _ _ _ ocstub-go-image-1.1.0.tar (Docker Images)

|

|_ _ _ocdns-pkg-1.1.0.0.0.tgz

|      |_ _ _ _ _ _ ocdns-bind-1.1.0.tgz(Helm Charts)

|      |_ _ _ _ _ _ ocdns-bind-image-1.1.0.tar (Docker Images)
```

11. The user can copy the tar file from here to their Kubernetes cluster where, they want to deploy ATS.

Deploying ATS in Kubernetes Cluster

To deploy ATS in Kubernetes Cluster:

1. Execute the following command to extract the tar file content:
   `tar -zxvf ocats-policy-tools-1.6.0.0.0.tgz`

   The output of this command is:

```
ocats-policy-pkg-1.6.0.0.0.tgz
ocamf-pkg-1.1.0.0.0.tgz
ocstub-pkg-1.1.0.0.0.tgz
ocdns-pkg-1.1.0.0.0.tgz
```

2. Go to the `ocats-policy-tools-1.6.0.0.0` folder and execute the following command to extract the final helm charts and docker images of ATS.
   `tar -zxvf ocats-policy-pkg-1.6.0.0.0.tgz`

The output of this command is:

```
ocats-policy-1.6.0.tgz
ocats-policy-images-1.6.0.tar
```

3. In your cluster, execute the given command to load the ATS image.
   `docker load --input ocats-policy-images-1.6.0.tar`
4. Execute the following commands to tag and push the ATS images

   ```
   docker tag ocats-policy:1.6.0 <registry>/ocats-policy:1.6.0
   docker push <registry>/ocats-policy:1.6.0
   ```

   Example:

   ```
   docker tag ocats-policy:1.6.0 localhost:5000/ocats-policy:1.6.0
   docker push localhost:5000/ocats-policy:1.6.0
   ```

5. Untar the helm charts, `ocats-policy-1.6.0.tgz`
   `tar -zxvf ocats-policy-1.6.0.tgz`
6. Update the registry name, image name and tag in the `ocats-policy/values.yaml` file as required.
   For this, you need to open the `values.yaml` file and update the `image.repository` and `image.tag`
7. ATS supports static port. By default, this feature is not available. To enable this feature:
   - In the **ocats-policy/values.yaml** file under service section, set the value of **staticNodePortEnabled** parameter as true and provide a valid nodePort value for **staticNodePort**.
   - A sample screen is given below:

   Figure 2-1 ocats-policy/values.yaml-service section

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

8. To enable service mesh feature:
   a. Under the **service** section of the **values.yaml** file, there is a parameter, '**serviceMeshCheck**'. By default, this feature is set to **false**. To get ASM support, set this parameter to **true**. A snippet of service section in the yaml file is shown below:

Figure 2-2 Service Mesh Check Enabled

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

b. If you do not enable ASM at global level for the namespace, then execute the following command to enable it before deploying the ATS.
`kubectl label --overwrite namespace <namespace_name> istio-injection=enabled`

Example: `kubectl label --overwrite namespace ocBSF istio-injection=enabled`

9. Deploy ATS using the updated helm charts (refer to **step 5** for helm charts).

> **Note:**
>
> You need to ensure that all the four components, 'ATS, go-Stub, dns-bind and CNPolicy are deployed in the same namespace.

Using Helm 2 `helm install ocats-policy --name <release_name> --namespace <namespace_name> -f ocats-policy/values.yaml`

Example: `helm install ocats-policy --name ocats --namespace ocBSF -f ocats-policy/values.yaml`

Using Helm 3 `helm3 install -name <release_name> ocats-policy-1.8.1.tgz --namespace <namespace_name> -f <values-yaml-file>`
Example: `helm3 install -name ocats ocats-policy-1.6.0.tgz --namespace ocBSF -f ocats-policy/values.yaml`

10. Execute the following command to verify ATS deployment.
`helm status <release_name>`

You should see the status as **Deployed** if the deployment is successful.

Deploying Stub Pod in Kubernetes Cluster

To deploy Stub Pod in Kubernetes cluster:

1. Go to the `ocats-policy-tools-1.6.0.0.0` folder and execute the command to extract the ocstub tar file content.
`tar -zxvf ocstub-pkg-1.1.0.0.0.tgz`

The output of this command is:

```
ocstub-go-1.1.0.tgz
ocstub-go-images-1.1.0.tar
```

2. In your cluster, execute the following command to load the STUB image
   `docker load --input ocstub-go-image-1.1.0.tar`
3. Execute the following commands to tag and push the STUB image

   ```
   docker tag ocstub-go:1.1.0 <registry>/ocstub-go:1.1.0
   ```

   ```
   docker push <registry>/ocstub-go:1.1.0
   ```
4. Untar the helm charts, `ocstub-go-1.1.0.tgz`.
   `tar -zxvf ocstub-go-1.1.0.tgz`
5. Update the registry name, image name and tag (if required) in the `ocstub-go/values.yaml` file as required.
   Open the **values.yaml** file and update the `image.repository` and `image.tag`
6. If required, change the `apiVersion` to `apps/v1` in the **ocstub-go/templates/deployment.yaml** file as shown below.
   `apiVersion: apps/v1`
7. Deploy Stub.
   Using Helm 2: `helm install ocstub-go --set service.name=<service> --name <name> --namespace <namespace_name> --set nameOverride=<App name> -f ocstub-go/values.yaml`

   Example:

   ```
   helm install ocstub-go --set service.name=nf1stub --name nf1stub --
   namespace
   ocbsf --set nameOverride=nf1stub -f ocstub-go/values.yaml
   ```

   Using Helm 3:`helm3 install -name <release_name> ocstub-go-1.1.0.tgz --set service.name=<stub-service-name> --namespace <namespace_name> --set nameOverride =<App name> -f <valuesyaml-file>`

   Example:

   ```
   helm3 install -name nf1stub ocstub-go-1.1.0.tgz --set
   service.name=nf1stub
   --namespace ocbsf --set nameOverride=nf1stub -f ocstub-go/
   values.yaml
   ```
8. Execute the following command to check the Stub deployment.
   `helm status <release_name>`

   You should see the status as **Running** and **Ready** for all the stubs if the deployment is successful.

NRF ATS Installation Procedure

The NRF ATS installation procedure covers three steps:

1. Locating and downloading ATS and Simulator Images
2. Preparing to deploy ATS and Stub Pod in Kubernetes Cluster
3. Deploying ATS and Stub Pod in Kubernetes Cluster

Locating and Downloading ATS Images

The steps to locate and download ATS Images are as follows:

1. The ATS Images are available on the OHC server. To download the ATS Images from OHC:
   a. Go to the URL, docs.oracle.com.
   b. Navigate to **Industries > Communications > Cloud Native Core**.
   c. Click the **Automated Testing Suite (ATS) Images** link to download the zip file.
   d. Unzip the **Images** folder to access all the ATS Images.
2. The `ocats-nrf` directory has following files:

```
ocats-nrf-tools-pkg-1.8.0.0.0.tgz
ocats-nrf-tools-pkg-1.8.0.0.0-README.txt
ocats-nrf-tools-pkg-1.8.0.0.0.tgz.sha256
ocats-nrf-custom-configtemplates-1.8.0.0.0.zip
ocats-nrf-custom-configtemplates-1.8.0.0.0-README.txt
```

3. The `ocats-nrf-tools-pkg-1.8.0.0.0-README.txt` file contains all the information required for the package.
4. The `ocats-nrf-tools-pkg-1.8.0.0.0.tgz` file has following images and charts packaged as tar files:

```
ocats-nrf-tools-pkg-1.8.0.0.0.tgz

      |

      |_ _ _ocats-nrf-pkg-1.8.0.0.0.tgz

      |        |_ _ _ _ _ _ ocats-nrf-1.8.0.tgz (Helm Charts)

      |        |_ _ _ _ _ _ ocats-nrf-image-1.8.0.tar (Docker
Images)

      |        |_ _ _ _ _ _ OCATS-NRF-Readme.txt

      |        |_ _ _ _ _ _ ocats-nrf-1.8.0.tgz.sha256

      |        |_ _ _ _ _ _ocats-nrf-image-1.8.0.tar.sha256

      |
```

```
      |

      |_ _ _ocstub-python-pkg-1.8.0.0.0.tgz

               |_ _ _ _ _ _ ocstub-python-1.8.0.tgz (Helm Charts)

               |_ _ _ _ _ _ ocstub-python-image-1.8.0.tar (Docker
Images)

               |_ _ _ _ _ _ OCSTUB-PYTHON-Readme.txt

               |_ _ _ _ _ _ ocstub-python-1.8.0.tgz.sha256

               |_ _ _ _ _ _ ocstub-python-image-1.8.0.tar.sha256
```

In addition to the above images and charts, the **ocats-nrf-custom-configtemplates-1.8.0.0.0.zip** file is also there in the same location. The **ocats-nrf-custom-configtemplates-1.8.0.0.0-README.txt** file contains the information about the content of this zip file. The content of the zip file is as follows:

```
ocats-nrf-custom-configtemplates-1.8.0.0.0.zip

      |

      |_ _ _ocats-nrf-custom-values.yaml (Custom values file for
installation)

      |

      |_ _ _ocats-nrf-custom-serviceaccount.yaml (Template to
create custom service account)

      |

      |_ _ _ocstub-python-custom-values.yaml (Custom values file
for stub installation)
```

5. The user can copy the tar file from here to the OCCNE/OCI/Kubernetes cluster where they want to deploy ATS.

Preparing to Deploy ATS and Stub Pod in Kubernetes Cluster

The steps to deploy ATS and Stub Pod in Kubernetes Cluster are as follows:

1. Execute the following command to extract tar file content.
   ```
   tar -xvf ocats-nrf-tools-pkg-1.8.0.0.0.tgz
   ```
   The output of this command is:

   ```
   ocats-nrf-pkg-1.8.0.0.0.tgz
   ocstub-python-pkg-1.8.0.0.0.tgz
   ```

2. Execute the following command to extract the final helm charts and docker images of ATS.
   ```
   tar -xvf ocats-nrf-pkg-1.8.0.0.0.tgz
   ```
   The output of this command is:

   ```
   ocats-nrf-image-1.8.0.tar
   ocats-nrf-1.8.0.tgz
   OCATS-NRF-Readme.txt
   ```

   > **Note:**
   >
   > The `OCATS-NRF-Readme.txt` file contains all the information required for the package.

3. Execute the following command to untar the ocstub package.
   ```
   tar -xvf ocstub-python-pkg-1.8.0.0.0.tgz
   ```
   The output of this command is:

   ```
   ocstub-python-image-1.8.0.tar
   ocstub-python-1.8.0.tgz
   OCSTUB-PYTHON-Readme.txt
   ```

4. Execute the following command to extract the content of the custom values file:
   ```
   unzip ocats-nrf-custom-configtemplates-1.8.0.0.0.zip
   ```
   The output of this command is:

   ```
   ocats-nrf-custom-values.yaml (Custom yaml file for deployment of
   OCATS-NRF)
   ocats-nrf-custom-serviceaccount.yaml (Custom yaml file for service
   account creation to help the customer if required)
   ocstub-python-custom-values.yaml (Custom yaml file for deployment
   of OCSTUB-PYTHON)
   ```

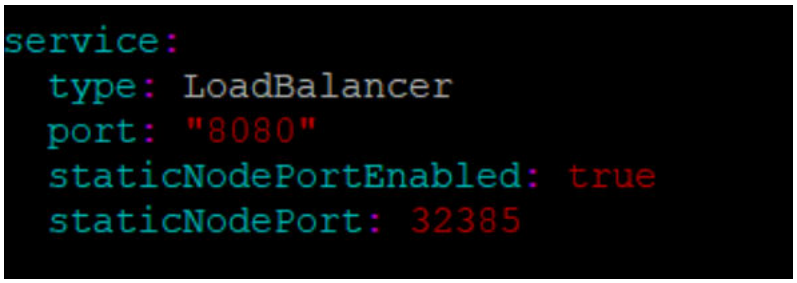
5. In your cluster, load the ATS docker image, '`ocats-nrf-image-1.8.0.tar`' and Stub docker image, '`ocstub-python-image-1.8.0.tar`' and push it to your registry.

   ```
   docker load -i ocats-nrf-image-1.8.0.tar
   docker tag ocats/ocats-nrf:1.8.0 <local_registry>/ocats/ocats-
   nrf:1.8.0
   docker push <local_registry>/ocats/ocats-nrf:1.8.0

   docker load -i ocstub-python-image-1.8.0.tar
   docker tag ocats/ocstub-python:1.8.0 <local_registry>/ocats/ocstub-
   python:1.8.0
   docker push <local_registry>/ocats/ocstub-python:1.8.0
   ```

6. Update the image name and tag in the `ocats-nrf-custom-values.yaml` and `ocstub-python-custom-values.yaml` file as required.
   For this, you need to open the `ocats-nrf-custom-values.yaml` and `ocstub-python-custom-values.yaml` file and update the `image.repository` and `image.tag`

7. ATS supports static port. By default, this feature is not available. To enable this feature:
   - In the **ocats-nrf-custom-values.yaml** file under service section, set the **staticNodePortEnabled** parameter value to 'true' and **staticNodePort** parameter value with valid nodePort.
   - A sample screen is given below:

Figure 2-3 ocats-nrf-custom-values.yaml - service section



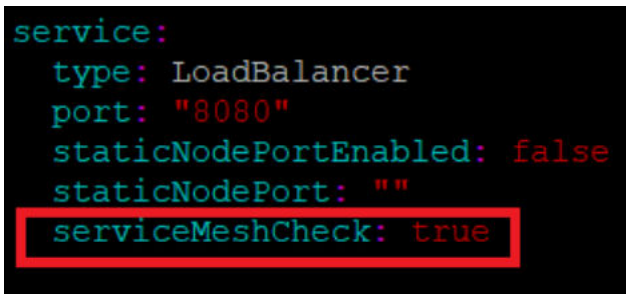
Enabling Service Mesh for ATS

> **Note:**
>
> This procedure is applicable only if you want to enable service mesh.

To enable service mesh for ATS, perform the following steps:

1. Under the **service** section of the **ocats-nrf-custom-values.yaml** file, set the **serviceMeshCheck** parameter **true**. By default, this parameter is set to false. A snippet of service section in the yaml file is given below:

Figure 2-4 Enabling Service Mesh



2. If the service mesh is not enabled at the global level for the namespace, execute the following command to enable it before deploying ATS.

```
kubectl label --overwrite namespace <namespace_name> istio-
injection=enabled
```

Example

```
kubectl label --overwrite namespace ocnrf istio-injection=enabled
```

3. Add the following annotation under the **lbDeployments** parameter of the global section in **ocats-nrf-custom-values.yaml** file. Sample is as follows:
`traffic.sidecar.istio.io/excludeInboundPorts: "8080"`

Figure 2-5 Sample Annotation

```
lbDeployments:
  labels: {}
  annotations:
    traffic.sidecar.istio.io/excludeInboundPorts: "8080"
```

Enabling NF FQDN Authentication Feature

> **Note:**
>
> **Perform below steps only if the NF FQDN Authentication feature is tested. Or, else proceed to the "Deploying ATS and Stub pod in K8s cluster" section.**

You must enable this feature while deploying Service Mesh. However, there is some change in the ATS deployment process, which is as follows:

1. Use previously unzipped file **"ocats-nrf-custom-serviceaccount.yaml"** to create a service account. Add the following annotation in the service-account file.
`"certificate.aspenmesh.io/customFields": '{ "SAN": { "DNS": [ "<NF-FQDN>" ] } }'`

Figure 2-6 Sample: Service Account File - Annotation

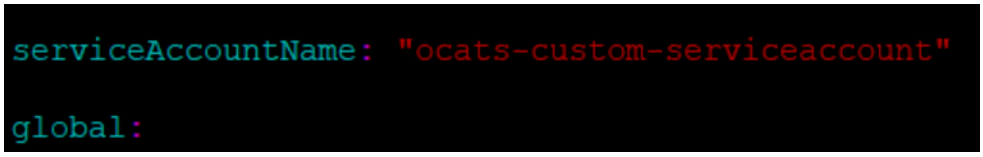
```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: ocats-custom-serviceaccount
  namespace: ocnrf
  annotations:
    "certificate.aspenmesh.io/customFields": '{ "SAN": { "DNS": [ "AMF.d5g.oracle.com" ]
```

> **Note:**
>
> "AMF.d5g.oracle.com" is the NF FQDN that you must provide in the serviceaccount DNS field.

2. Execute the following command to create a service account:
`kubectl apply -f ocats-nrf-custom-serviceaccount.yaml`

3. After creating the service account, update the service account name in the **ocats-nrf-custom-values.yaml** file as shown below:

Figure 2-7 Updating Service Account Name



4. Deploy ATS using helm2 or helm3 commands shared in the **Deploying ATS and Stub Pod in Kubernetes Cluster** section.

Deploying ATS and Stub Pod in Kubernetes Cluster

> **Note:**
>
> **It is important to ensure that all the three components; ATS, Stub and NRF are in the same namespace.**

You need two Stubs for the NRF tests to be executed. The service name for the stubs should be **notify-stub-service** and **notify-stub-service02**.

ATS and Stub supports Helm2 and Helm3 for deployment.

If the namespace does not exists, execute the following command to create a namespace:

```
kubectl create namespace ocnrf
```

Using Helm 2 for ATS:

```
helm install ocats-nrf-1.8.0.tgz --name <release_name> --namespace
<namespace_name> -f <values-yaml-file>
```

Example:

```
helm install ocats-nrf-1.8.0.tgz --name ocats --namespace ocnrf -f
ocats-nrf-custom-values.yaml
```

Using Helm 2 for Stubs:

```
helm install ocstub-python-1.8.0.tgz --set service.name=<stub-service-
name>
--name <release_name> --namespace <namespace_name> -f <values-yaml-file>
```

Example:

```
helm install ocstub-python-1.8.0.tgz --set service.name=notify-stub-
service --name
ocstub --namespace ocnrf -f ocstub-python-custom-values.yaml
helm install ocstub-python-1.8.0.tgz --set service.name=notify-stub-
service02 --name
ocstub1 --namespace ocnrf -f ocstub-python-custom-values.yaml
```

Using Helm 3 for ATS:

```
helm3 install -name <release_name> ocats-nrf-1.8.0.tgz --namespace
<namespace_name> -f <values-yaml-file>
```

Example:
```
helm3 install -name ocats ocats-nrf-1.8.0.tgz --namespace ocnrf -f
ocats-nrf-custom-values.yaml
```

Using Helm 3 for Stubs:

```
helm3 install -name <release_name> ocstub-python-1.8.0.tgz --set
service.name=<stub-service-name> --namespace <namespace_name> -f
<values-yaml-file>
```

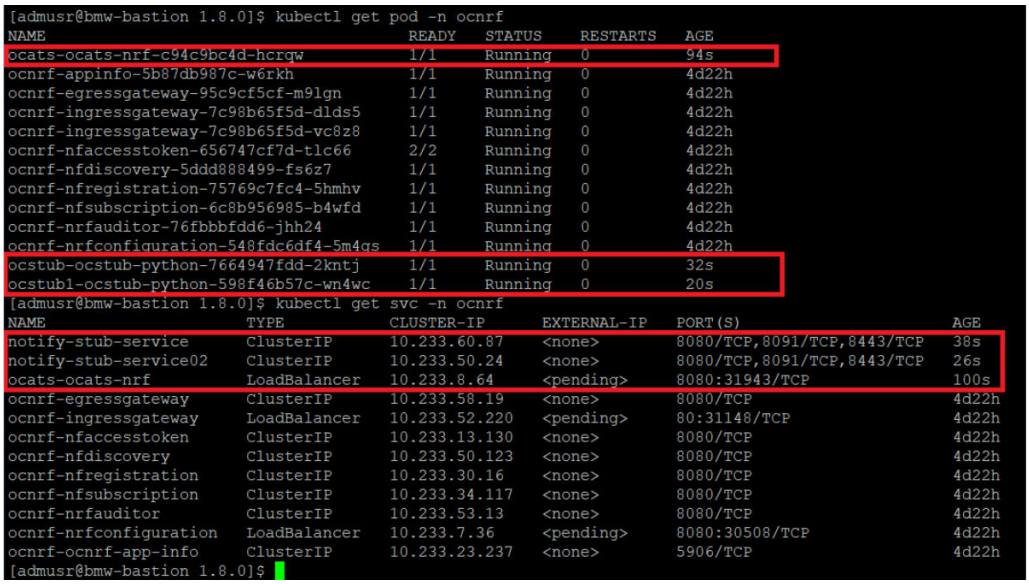
Example:
```
helm3 install -name ocstub ocstub-python-1.8.0.tgz --set
service.name=notify-stub-service --namespace ocnrf -f ocstub-python-
custom-values.yaml
helm3 install -name ocstub1 ocstub-python-1.8.0.tgz --set
service.name=notify-stub-service02 --namespace ocnrf -f ocstub-python-
custom-values.yaml
```

Execute the following command to verify ATS deployment.
```
helm status <release_name>
```

Once ATS and Stub are deployed, execute the following command to check the pod and service deployment.

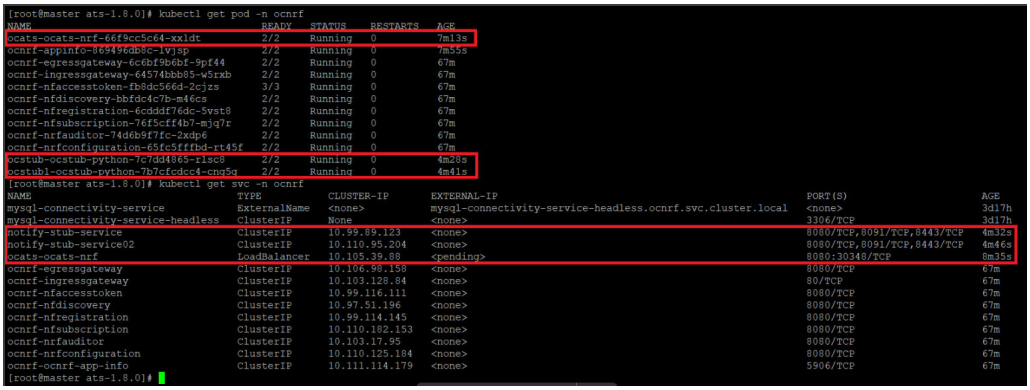
Checking Pod Deployment:
```
kubectl get pod -n ocnrf
```
Checking Service Deployment:
```
kubectl get service -n ocnrf
```

Figure 2-8 Checking Pod and Service Deployment without Service Mesh



If ATS is deployed with side car of service mesh, you need to ensure that both ATS and Stub pods have 2 containers in ready state and shows **"2/2"**. A sample screen is shown below:

Figure 2-9 ATS and Stub Deployed with Service Mesh



NSSF ATS Installation Procedure

The NSSF ATS installation procedure covers two steps:

1. Locating and downloading ATS and Simulator Images
2. Deploying ATS and Stub Pod in Kubernetes Cluster as per NSSF

Locating and Downloading ATS Images

1. The ATS Images are available on the OHC server. To download the ATS Images from OHC:

a. Go to the URL, docs.oracle.com
b. Navigate to **Industries** > **Communications** > **Cloud Native Core**
c. Click the **Automated Testing Suite (ATS) Images** link to download the zip file.
d. Unzip the **Images** folder to access all the ATS Images.

2. The `ocats-nssf` directory has the following files:
   - `ocats-nssf-tools-pkg-1.4.0.0.tgz`
   - `ocats-nssf-tools-pkg-1.4.0.0-README.txt`

> **Note:**
> The `ocats-nssf-tools-pkg-1.4.0.0-README.txt` file contains all the information required for the package.

3. The `ocats-nssf-tools-pkg-1.4.0.0-README.txt` file contains all the information required for the package.
4. The `ocats-nssf-tools-pkg-1.4.0.0.tgz` file has the following images and charts packaged as tar files:

```
ocats-nssf-tools-pkg-1.4.0.0.tgz
|
|_ _ _ocats-nssf-pkg-1.4.0.0.tgz
| |_ _ _ _ _ _ ocats-nssf-1.4.tgz (Helm Charts)
| |_ _ _ _ _ _ ocats-nssf-image-1.4.tar (Docker Images)
| |_ _ _ _ _ _ Readme.txt
```

5. The user can copy the tar file from here and copy in their OCCNE/OCI/Kubernetes cluster where they want to deploy ATS.

Deploying ATS in Kubernetes Cluster

The steps to deploy ATS in Kubernetes Cluster are as follows:

1. Execute the following command to extract tar file content:
   `tar -xvf ocats-nssf-tools-pkg-1.4.0.0.tgz`

   The output of this command is:

   `ocats-nssf-pkg-1.4.0.0.tgz`
2. Execute the following command to extract final helm charts and docker images of ATS:
   `tar -xvf ocats-nssf-pkg-1.4.0.0.tgz`

   The output of this command is:

   `ocats-nssf-image-1.4.tar ocats-nssf-1.4.tgz`

   `Readme.txt`
3. In your cluster, load the ATS image, '`ocats-nssf-image-<version>.tar`' and push to your registry.
   `docker load -i ocats-nssf-image-<version>.tar`

a. Execute the following command to grep the image.
`docker images | grep ocats-nssf`

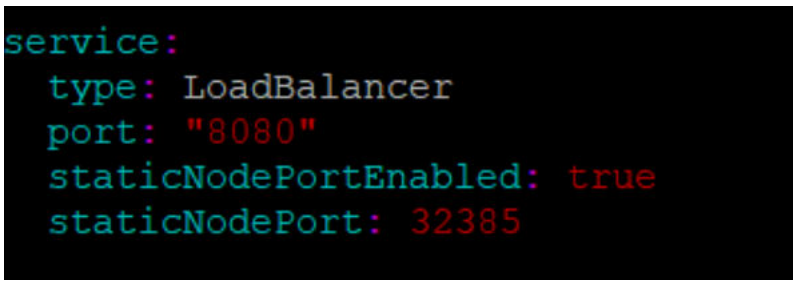
b. Copy the Image ID from the output of the grep command and change the tag to your registry.
Example:

```
docker tag <Image_ID> <your-registry-name/ocats-nssf:<tag>>
```

```
docker push <your-registry-name/ocats-nssf:<tag>>
```

4. Untar the helm charts, `ocats-nssf-<version>.tgz`
`tar -xvf ocats-nssf-<version>.tgz`
5. Update the image name and tag in the `ocats-nssf/values.yaml` file as required. For this, you need to open the `values.yaml` file and update the `image.repository` and `image.tag`.
6. ATS supports static port. By default, this feature is not available. To enable this feature:
   - In the **ocats-nssf/values.yaml** file under service section, set the value of **staticNodePortEnabled** parameter as true and provide a valid nodePort value for **staticNodePort**.
   - A sample screen is given below:

Figure 2-10 ocats-nssf/values.yaml - service section



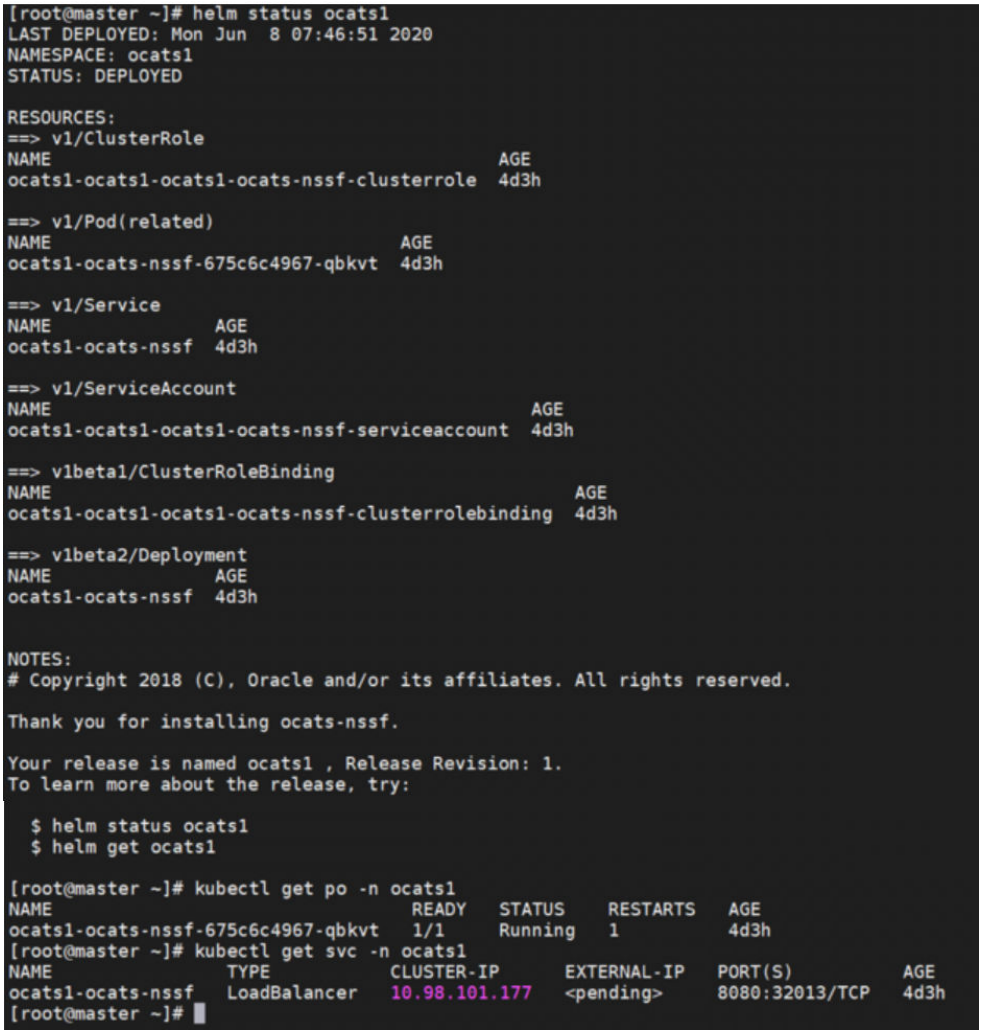
7. Deploy ATS using the updated helm charts after performing the previous **step 5**.
`helm install ocats-nssf --name <release_name> --namespace <namespace_name> -f ocats-nssf/values.yaml`

   Example: `helm install ocats-nssf --name ocats --namespace ocnssf -f ocats-nssf/values.yaml`

   If this command returns an error like, `<Error: validation failed: unable to recognize "": no matches for kind "Deployment" in version "apps/v1beta2">` then, open the `template/deployment.yml` file and change the apiVersoin to **apiVersion: apps/v1**.
8. Execute the following command to verify the ATS deployment:
`helm status <release_name>`

   A sample screen showing ATS Helm release is given below:

Figure 2-11 ATS Helm Release



Policy ATS Installation Procedure

The Policy ATS installation procedure covers two steps:

1. Locating and downloading the ATS images.
2. Deploying ATS images.

This includes installation of nine stubs (nf1stub, nf11stub, nf12stub, nf2stub, nf21stub, nf22stub, nf3stub, nf31stub, nf32stub), ATS, and ocdns-bind stub in Policy's namespace (ocpcf). The release of ATS supports incluster deployment of Policy and ATS with both TLS (server side) enabled and disabled mode.

> **Note:**
>
> Restart the Nrf-client pod of Policy for UDR and CHF discovery as part of each test case.

Downloading ATS Image

To download the ATS Image from MOS:

1. Login to My Oracle Support with your credentials.
2. Select **Patches and Updates** tab to locate the patch.
3. In **Patch Search** window, click **Product or Family (Advanced)**.
4. Enter *Oracle Communications Cloud Native Core - 5G* in **Product** field, select *Oracle Communications Cloud Native Core Policy 1.8.0.0.0* from **Release** drop-down.
5. Click on **Search**. The **Patch Advanced Search Results** displays a list of releases.
6. Select the required patch from the search results. The Patch Details window opens.
7. Click **Download**. File Download window appears.
8. Click the **<p********_<release_number>_Tekelec>.zip** file to downlaod the CNC Policy ATS package file.
9. Untar the zip file to access all the ATS Images.
10. The `ocats-policy-tools-1.8.1.0.0.tgz` file has following images and charts packaged as tar files:

```
ocats-policy-tools-1.8.1.0.0.tgz

|

|_ _ _ocats-policy-pkg-1.8.1.0.0.tgz

|      |_ _ _ _ _ _ ocats-policy-1.8.1.tgz (Helm Charts)

|      |_ _ _ _ _ _ ocats-policy-images-1.8.1.tar (Docker Images)

|

|_ _ _ocstub-pkg-1.1.0.0.0.tgz

|      |_ _ _ _ _ _ ocstub-go-1.1.0.tgz(Helm Charts)

|      |_ _ _ _ _ _ ocstub-go-image-1.1.0.tar (Docker Images)

|

|_ _ _ocdns-pkg-1.1.0.0.0.tgz

|      |_ _ _ _ _ _ ocdns-bind-1.1.0.tgz(Helm Charts)

|      |_ _ _ _ _ _ ocdns-bind-image-1.1.0.tar (Docker Images)
```

11. The user can copy the tar file from here to their Kubernetes cluster where, they want to deploy ATS.

Deploying ATS in Kubernetes Cluster

To deploy ATS in Kubernetes Cluster:

1. Execute the following command to extract the tar file content:
   `tar -zxvf ocats-policy-tools-1.8.1.0.0.tgz`

   The output of this command is:

   ```
   ocats-policy-pkg-1.8.1.0.0.tgz
   ocstub-pkg-1.1.0.0.0.tgz
   ocdns-pkg-1.1.0.0.0.tgz
   ```

2. Go to the `ocats-policy-tools-1.8.1.0.0` folder and execute the following command to extract the final helm charts and docker images of ATS.
   `tar -zxvf ocats-policy-pkg-1.8.1.0.0.tgz`

   The output of this command is:

   ```
   ocats-policy-1.8.1.tgz
   ocats-policy-images-1.8.1.tar
   ```

3. In your cluster, execute the given command to load the ATS image.
   `docker load --input ocats-policy-images-1.8.1.tar`
4. Execute the following commands to tag and push the ATS images

   ```
   docker tag ocats-policy:1.8.1 <registry>/ocats-policy:1.8.1
   docker push <registry>/ocats-policy:1.8.1
   ```

   Example:

   ```
   docker tag ocats-policy:1.8.1 localhost:5000/ocats-policy:1.8.1
   docker push localhost:5000/ocats-policy:1.8.1
   ```

5. Untar the helm charts, `ocats-policy-1.8.1.tgz`
   `tar -zxvf ocats-policy-1.8.1.tgz`
6. Update the registry name, image name and tag in the `ocats-policy/values.yaml` file as required.
   For this, you need to open the `values.yaml` file and update the `image.repository` and `image.tag`
7. ATS supports static port. By default, this feature is not available. To enable this feature:
   - In the **ocats-policy/values.yaml** file under service section, set the value of **staticNodePortEnabled** parameter as true and provide a valid nodePort value for **staticNodePort**.
   - A sample screen is given below:

Figure 2-12 ocats-policy/values.yaml-service section

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

8. To enable service mesh feature:
   a. Under the **service** section of the **values.yaml** file, there is a parameter, '**serviceMeshCheck**'. By default, this feature is set to **false**. To get ASM support, set this parameter to **true**. A snippet of service section in the yaml file is shown below:

Figure 2-13 Service Mesh Check Enabled

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

   b. If you do not enable ASM at global level for the namespace, then execute the following command to enable it before deploying the ATS.
   `kubectl label --overwrite namespace <namespace_name> istio-injection=enabled`

   Example: `kubectl label --overwrite namespace ocpcf istio-injection=enabled`

9. Deploy ATS using the updated helm charts (refer to **step 5** for helm charts).

> **Note:**
>
> You need to ensure that all the four components, 'ATS, go-Stub, dns-bind and CNPolicy are deployed in the same namespace.

Using Helm 2 `helm install ocats-policy --name <release_name> --namespace <namespace_name> -f ocats-policy/values.yaml`

Example: `helm install ocats-policy --name ocats --namespace ocpcf -f ocats-policy/values.yaml`

Using Helm 3 `helm3 install -name <release_name> ocats-policy-1.8.1.tgz --namespace <namespace_name> -f <values-yaml-file>`
Example: `helm3 install -name ocats ocats-policy-1.8.1.tgz --namespace ocpcf -f ocats-policy/values.yaml`

10. Execute the following command to verify ATS deployment.
    `helm status <release_name>`

Figure 2-14 Verifying ATS Deployment in Policy Namespace

Deploying Stub Pod in Kubernetes Cluster

To deploy Stub Pod in Kubernetes cluster:

1. Go to the `ocats-policy-tools-1.8.1.0.0` folder and execute the command to extract the ocstub tar file content.
   `tar -zxvf ocstub-pkg-1.1.0.0.0.tgz`

   The output of this command is:

   ```
   ocstub-go-1.1.0.tgz
   ocstub-go-images-1.1.0.tar
   ```

> **Note:**
>
> To deploy additional stubs required for session, retry feature validation:
>
> - nf11stub, nf12stub → Alternate FQDN for nf1stub
> - nf21stub, nf22stub → Alternate FQDN for nf2stub
> - nf31stub, nf32stub → Alternate FQDN for nf3stub

2. In your cluster, execute the following command to load the STUB image
   `docker load --input ocstub-go-image-1.1.0.tar`
3. Execute the following commands to tag and push the STUB image

   ```
   docker tag ocstub-go:1.1.0 <registry>/ocstub-go:1.1.0
   ```

   ```
   docker push <registry>/ocstub-go:1.1.0
   ```

4. Untar the helm charts, `ocstub-go-1.1.0.tgz`.
   `tar -zxvf ocstub-go-1.1.0.tgz`

5. Update the registry name, image name and tag (if required) in the `ocstub-go/values.yaml` file as required.
   Open the **values.yaml** file and update the `image.repository` and `image.tag`
6. If required, change the `apiVersion` to `apps/v1` in the **ocstub-go/templates/deployment.yaml** file as shown below.
   `apiVersion: apps/v1`
7. Deploy Stub.
   Using Helm 2: `helm install ocstub-go --set service.name=<service> --name <name> --namespace <namespace_name> -f ocstub-go/values.yaml`

   Example:

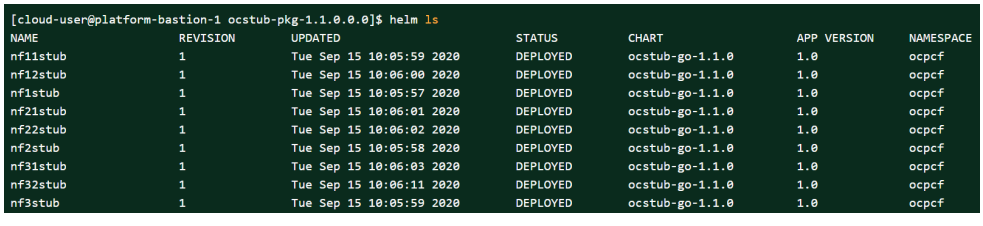
```
helm install ocstub-go --set service.name=nf1stub --name nf1stub --
namespace
ocpcf -f ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf2stub --name nf2stub --
namespace ocpcf -f
ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf3stub --name nf3stub --
namespace ocpcf -f
ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf11stub --name nf11stub
--namespace ocpcf -f
ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf12stub --name nf12stub
--namespace ocpcf -f
ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf21stub --name nf21stub
--namespace ocpcf -f
ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf22stub --name nf22stub
--namespace ocpcf -f
ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf31stub --name nf31stub
--namespace ocpcf -f
ocstub-go/values.yaml

helm install ocstub-go --set service.name=nf32stub --name nf32stub
--namespace ocpcf -f
ocstub-go/values.yaml
```

   Using Helm 3:`helm3 install -name <release_name> ocstub-go-1.1.0.tgz --set service.name=<stub-service-name> --namespace <namespace_name> -f <valuesyaml-file>`

Example:

```
helm3 install -name nf1stub ocstub-go-1.1.0.tgz --set
service.name=nf1stub
--namespace ocpcf -f ocstub-go/values.yaml

helm3 install -name nf2stub ocstub-go-1.1.0.tgz --set
service.name=nf2stub --namespace
ocpcf -f ocstub-go/values.yaml

helm3 install -name nf3stub ocstub-go-1.1.0.tgz --set
service.name=nf3stub --namespace
ocpcf -f ocstub-go/values.yaml

helm3 install -name nf3stub ocstub-go-1.1.0.tgz --set
service.name=nf11stub --namespace
ocpcf -f ocstub-go/values.yaml

helm3 install -name nf3stub ocstub-go-1.1.0.tgz --set
service.name=nf12stub --namespace
ocpcf -f ocstub-go/values.yaml

helm3 install -name nf3stub ocstub-go-1.1.0.tgz --set
service.name=nf21stub --namespace
ocpcf -f ocstub-go/values.yaml

helm3 install -name nf3stub ocstub-go-1.1.0.tgz --set
service.name=nf22stub --namespace
ocpcf -f ocstub-go/values.yaml

helm3 install -name nf3stub ocstub-go-1.1.0.tgz --set
service.name=nf31stub --namespace
ocpcf -f ocstub-go/values.yaml

helm3 install -name nf3stub ocstub-go-1.1.0.tgz --set
service.name=nf32stub --namespace
ocpcf -f ocstub-go/values.yaml
```

Figure 2-15 Stub - Checking Helm Status



```
[cloud-user@platform-bastion-1 ocstub-pkg-1.1.0.0.0]$ helm ls
NAME            REVISION    UPDATED                     STATUS      CHART             APP VERSION  NAMESPACE
nf11stub        1           Tue Sep 15 10:05:59 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf12stub        1           Tue Sep 15 10:06:00 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf1stub         1           Tue Sep 15 10:05:57 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf21stub        1           Tue Sep 15 10:06:01 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf22stub        1           Tue Sep 15 10:06:02 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf2stub         1           Tue Sep 15 10:05:58 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf31stub        1           Tue Sep 15 10:06:03 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf32stub        1           Tue Sep 15 10:06:11 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
nf3stub         1           Tue Sep 15 10:05:59 2020    DEPLOYED    ocstub-go-1.1.0   1.0          ocpcf
```

8. Similarly, install all other stubs.
9. Execute the following command to check the Stub deployment.
   `helm status <release_name>`

A sample screen showing stubs deployment is given below:

Figure 2-16 Stubs After Installation

```
[cloud-user@platform-bastion-1 ocstub-pkg-1.1.0.0.0]$ kubectl get po -n ocpcf
NAME                                          READY   STATUS    RESTARTS   AGE
nf11stub-ocstub-go-66449ddb94-qg2j9           1/1     Running   0          19h
nf12stub-ocstub-go-6b8575487-l8pxv            1/1     Running   0          19h
nf1stub-ocstub-go-5ff485954c-prc2x            1/1     Running   0          19h
nf21stub-ocstub-go-56cf5b77fc-x8wkr           1/1     Running   0          19h
nf22stub-ocstub-go-547dfdf476-4j2sn           1/1     Running   0          19h
nf2stub-ocstub-go-6fb6f786d6-bc9fr            1/1     Running   0          19h
nf31stub-ocstub-go-c6c6d5584-5m48z            1/1     Running   0          19h
nf32stub-ocstub-go-848dfc7757-q797z           1/1     Running   0          19h
nf3stub-ocstub-go-6cb769ccd9-4fv9b            1/1     Running   0          19h
```

Figure 2-17 Policy Namespace

```
[cloud-user@platform-bastion-1 ocstub-pkg-1.1.0.0.0]$ kubectl get po -n ocpcf
NAME                                          READY   STATUS    RESTARTS   AGE
funocats-ocats-policy-54f9469654-8kggc        1/1     Running   0          3h34m
ocpcf-appinfo-6659cb6bbf-w86pv                1/1     Running   0          2d1h
ocpcf-oc-binding-7c99dccdcf-4z5s6             1/1     Running   0          2d1h
ocpcf-oc-diam-gateway-0                       1/1     Running   0          22h
ocpcf-occnp-alternate-route-77f587fdb6-l8w2b  1/1     Running   0          119m
ocpcf-occnp-alternate-route-77f587fdb6-zd96s  1/1     Running   0          124m
ocpcf-occnp-config-server-74747fd78c-54f2h    1/1     Running   0          2d1h
ocpcf-occnp-egress-gateway-65df684f99-rts7l   1/1     Running   0          2d1h
ocpcf-occnp-ingress-gateway-5c6f4dd876-2xwnp  1/1     Running   0          2d1h
ocpcf-occnp-nrf-client-nfdiscovery-85f5b8cfff-28sqw  1/1  Running  0        2d1h
ocpcf-occnp-nrf-client-nfmanagement-5c54974655-t29jd  1/1  Running  0       36m
ocpcf-ocpm-audit-service-5f96785f4d-6bbkm     1/1     Running   0          2d1h
ocpcf-ocpm-cm-service-69cdff54cc-bd928        1/1     Running   0          2d1h
ocpcf-ocpm-pre-5f6f447c47-6zw42               1/1     Running   0          2d1h
ocpcf-ocpm-pre-test-567c5fc84c-8sprm          1/1     Running   0          2d1h
ocpcf-ocpm-queryservice-84c4487c99-4qrp4      1/1     Running   0          2d1h
ocpcf-pcf-amservice-d45c7ff67-7gmc9           1/1     Running   0          2d1h
ocpcf-pcf-diam-connector-744654759c-vrgkt     1/1     Running   0          2d1h
ocpcf-pcf-smservice-7f9c5f58db-22wp8          1/1     Running   0          2d1h
ocpcf-pcf-ueservice-6546f54ccf-lpkf4          1/1     Running   0          2d1h
ocpcf-pcf-userservice-55956bc5b9-7vwjr        1/1     Running   0          2d1h
ocpcf-pcrf-core-7cd8b688bc-j6q28              1/1     Running   0          2d1h
ocpcf-performance-5f49787486-29gqg            1/1     Running   0          2d1h
nf11stub-ocstub-go-66449ddb94-qg2j9           1/1     Running   0          19h
nf12stub-ocstub-go-6b8575487-l8pxv            1/1     Running   0          19h
nf1stub-ocstub-go-5ff485954c-prc2x            1/1     Running   0          19h
nf21stub-ocstub-go-56cf5b77fc-x8wkr           1/1     Running   0          19h
nf22stub-ocstub-go-547dfdf476-4j2sn           1/1     Running   0          19h
nf2stub-ocstub-go-6fb6f786d6-bc9fr            1/1     Running   0          19h
nf31stub-ocstub-go-c6c6d5584-5m48z            1/1     Running   0          19h
nf32stub-ocstub-go-848dfc7757-q797z           1/1     Running   0          19h
nf3stub-ocstub-go-6cb769ccd9-4fv9b            1/1     Running   0          19h
ocdns-ocdns-bind-86888f75cf-kv64w             1/1     Running   0          19h
```

Deploying DNS Stub in Kubernetes Cluster

> **Note:**
>
> Please make sure that the sufficient resource requests and limit is configured for DNS Stub. Set the resource request and limit values in the **resources** section in the **values.yaml** file as follows:

```
resources: {}
  # We usually recommend not to specify default resources and
to leave this as a conscious
  # choice for the user. This also increases chances charts
run on environments with little
  # resources, such as Minikube. If you do want to specify
resources, uncomment the following
  # lines, adjust them as necessary, and remove the curly
braces after 'resources:'.  # limits:
  #  cpu: 1000m
  #  memory: 1024Mi
  # requests:
  #  cpu: 500m
  #  memory: 500Mi
```

To deploy DNS stub in Kubernetes cluster:

1. Go to the **ocats-policy-tools-1.8.1.0.0** folder and execute the following command:
   `tar -zxvf ocdns-pkg-1.1.0.0.0.tgz`

   The output is shown below:

   Figure 2-18 Untar DNS Package

   ```
   [cloud-user@platform-bastion-1 ocdns-pkg-1.1.0.0.0]$ ls -ltrh
   total 211M
   -rw-------. 1 cloud-user cloud-user 211M Sep 14 14:49 ocdns-bind-image-1.1.0.tar
   -rw-r--r--. 1 cloud-user cloud-user 2.9K Sep 14 14:49 ocdns-bind-1.1.0.tgz
   ```

2. In your cluster, execute the following command to load the DNS Stub image:
   `docker load --input ocdns-bind-image-1.1.0.tar`
3. Execute the following command to tag and push the DNS stub to the registry:
   `docker tag ocdns-bind:1.1.0 localhost:5000/ocdns-bind:1.1.0`

   `docker push localhost:5000/ocdns-bind:1.1.0`
4. Execute the following command to untar the helm charts (ocdns-bind-1.1.0.tgz):
   `tar -zxvf ocdns-bind-1.1.0.tgz`
5. Update the registry name, image name and tag (if required) in the `ocdns-bind/values.yaml` file as required. Open the **values.yaml** file and update the `image.repository` and `image.tag`

6. Execute the following command to install DNS Stub:

```
helm2 :
[cloud-user@platform-bastion-1 ocdns-bind]$ helm install ocdns-
bind-1.1.0.tgz --name
ocdns --namespace ocpcf -f ocdns-bind/values.yaml

helm3 :
[cloud-user@platform-bastion-1 ocdns-bind]$ helm3 install -name
ocdns
ocdns-bind-1.1.0.tgz --namespace ocpcf -f ocdns-bind/values.yaml
```

7. Execute the following command to capture the cluster name of the pcf deployment, namespace where nfstubs are deployed and cluster IP of DNS Stub.
`kubectl get svc -n ocpcf | grep dns`

Figure 2-19 DNS Stub Cluster IP

```
[cloud-user@platform-bastion-1 ocdns-pkg-1.1.0.0.0]$ kubectl get svc -n ocpcf | grep dns
NAME      TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)               AGE
ocdns     ClusterIP      10.233.11.45    <none>          53/UDP,6236/TCP       19h
```

> **Note:**
>
> This information is required to configure DNS stub.

Figure 2-20 Cluster Name

```
[cloud-user@platform-bastion-1 ocdns-pkg-1.1.0.0.0]$ kubectl -n kube-system get configmap kubeadm-config -o yaml | grep clusterName
    clusterName: platform
```

Replacing PCF Service Ports in ATS

> **Note:**
>
> After starting/restarting the ATS Pod, you have to execute the same script to update the ports.

To replace the PCF Service Ports:

1. Create a script called **replace_port.sh** on the server from where you are executing the *kubectl* commands.
2. Add the following content to the script:

```
#!/bin/bash
NAMESPACE=${NAMESPACE}
atspod=$(kubectl get pod -n ${NAMESPACE} | grep ocats | awk
'{ print $1 }')
config_mgmt_svc=$(kubectl get svc -n ${NAMESPACE} | grep config-
```

```
mgmt | awk '{ print $1 }')
config_mgmt_port=$(kubectl get svc -n ${NAMESPACE} $
{config_mgmt_svc} -o jsonpath={.spec.ports[].port})

config_server_svc=$(kubectl get svc -n ${NAMESPACE} | grep config-
server | awk '{ print $1 }')
config_server_port=$(kubectl get svc -n ${NAMESPACE} $
{config_server_svc} -o jsonpath={.spec.ports[].port})

echo -e "Please see below Environment variables"
echo -e "NAMESPACE: ${NAMESPACE}"
echo -e "ATS pod: ${atspod}"
echo -e "config-mgmt port: ${config_mgmt_port}"
echo -e "config-server port: ${config_server_port}"

# This Step will login to ATS pod and dynamically replace the port
for 'config-mgmt'
and 'config-server' services
kubectl exec -it ${atspod} -n ${NAMESPACE} -- bash -c "egrep -lRZ
'config-mgmt'
/var/lib/jenkins/ocpcf_tests/features/ | xargs -0 -l sed -i -e 's/
config-mgmt.
*$/config-mgmt '"$config_mgmt_port"'/g' && \egrep -lRZ 'config-
server'
/var/lib/jenkins/ocpcf_tests/features/ | xargs -0 -l sed -i -e
's/config-server.*$/config-server '"$config_server_port"'/g'"
echo -e "Successfully updated config-mgmt port to $
{config_mgmt_port} and config-server port to ${config_server_port}
in all the feature files"
```

3. Provide execute permission as follows:
   `chmod +x replace_port.sh`
4. Execute the following command to replace the PCF Service Port:
   `NAMESPACE=<PCF Namespace> ./replace_port.sh`

   Example: `NAMESPACE=ocpcf ./replace_port.sh`

SCP ATS Installation Procedure

The SCP ATS installation procedure covers two steps:

1. Locating and downloading the ATS images.
2. Deploying ATS images.

Locating and Downloading ATS Images

To locate and download ATS Images:

1. Download the ATS Images that are available on the OHC server:
   a. Go to the URL, docs.oracle.com
   b. Navigate to **Industries > Communications > Cloud Native Core**
   c. Click the **Automated Testing Suite (ATS) Images** link to download the zip file.

d. Unzip the **Images** folder to access all the ATS Images.

2. The `ocats-scp` directory has a following files:

```
ocats-scp-pkg-1.8.0.0.0.tgz
ocats-scp-pkg-1.8.0.0.0-readme.txt
ocats-scp-custom-configtemplates-1.8.0.0.0.zip
ocats-scp-custom-configtemplates-1.8.0.0.0-readme.txt
```

> **Note:**
>
> The `ocats-scp-custom-configtemplates-1.8.0.0.0-readme.txt` file contains all the information required for the package.

The `ocats-scp-pkg-1.8.0.0.0.tgz` file has following images and charts packaged as tar files:

```
ocats-scp-pkg-1.8.0.0.0.tgz

      |

      |_ _ _ocats-scp-pkg-1.8.0.0.0.tgz

      |        |_ _ _ _ _ _ ocats-scp-1.8.0.tgz (Helm Charts)

      |        |_ _ _ _ _ _ ocats-scp-images-1.8.0.tar (Docker
Images)

      |        |_ _ _ _ _ _ Readme.txt
```

The `ocats-scp-custom-configtemplates-1.8.0.0.0.zip` file has following images and charts packaged as tar files:

```
ocats-scp-custom-configtemplates-1.8.0.0.0.zip

            |_ _ _ _ _ _ ocats-scp-custom-serviceaccount-1.8.0.yaml
(Template to create custom service account)

              |_ _ _ _ _ _ ocats-scp-values-1.8.0.yaml (Custom
values file for installation)
```

The user can copy the tar file from here to their kubernetes cluster where, they want to deploy ATS.

Deploying ATS in Kuberbetes Cluster

To deploy ATS in Kubernetes Cluster:

> **Note:**
>
> Deploy ATS and SCP in the same namespace.

> **Note:**
>
> ATS is deployed with role binding by default instead of cluster role binding.

1. Execute the following command to extract the tar file content.
   `tar -xvf ocats-scp-pkg-1.8.0.0.0.tgz`

   The output of this command is:

   ```
   ocats-scp-1.8.0.tgz
   ocats-scp-images-1.8.0.tar
   Readme.txt
   ```

   The `ocats-scp-images-1.8.0.tar` file contains ocats-scp:1.8.0 (ATS Image) and ocats-gostub:1.8.0 (stub image).
2. In your cluster, execute the given command to load the ATS image and then, push it to your registry.
   `docker load --input ocats-scp-images-1.8.0.tar`
3. Execute the following command to extract the zip file content.

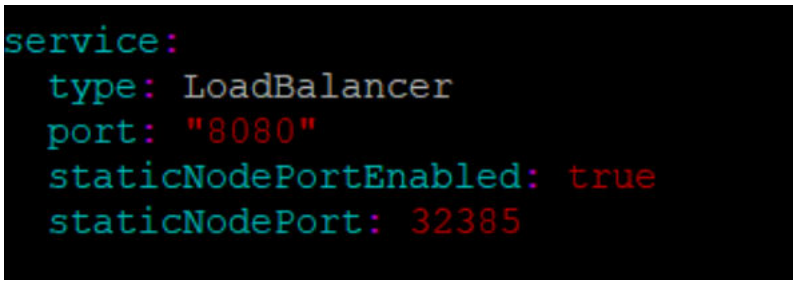
   ```
   Unzip "ocats-scp-custom-configtemplates-1.8.0.0.0.zip"
   ```

   The output of this command is:

   ```
   ocats-scp-values-1.8.0.yaml
   ocats-scp-custom-serviceaccount-1.8.0.yaml
   ```

4. Update the image name and tag in the **ocats-scp-values-1.8.0.yaml** file as required.
   For this, you need to open the `ocats-scp-values-1.8.0.yaml` file and update the `image.repository` and `image.tag`
5. ATS supports static port. By default, this feature is not available. To enable this feature:
   - In the **ocats-scp-values-1.8.0.yaml** file under service section, set the value of **staticNodePortEnabled** parameter as true and provide a valid nodePort value for **staticNodePort**.
   - A sample screen is given below:

Figure 2-21 ocats-scp-values-1.8.0.yaml- service section



> **Note:**
>
> You can enable static node port at the time of deployment.

- Add an appropriate value for the **serviceMeshCheck** parameter. Its value depends on whether ATS needs to be executed with aspen mesh or not.

6. Update the **lbDeployments** section of the helm deployment file in SCP ATS with the following annotations, wherein
   - 8091 port is added to fetch soothsayer pod metrics
     `traffic.sidecar.istio.io/excludeOutboundPorts: "8091"`

> **Note:**
>
> **This point is applicable only if you are planning to test ATS with service mesh. Also, do not modify this port.**

7. Execute the following command to deploy ATS.
   Using Helm 2: `helm install ocats-scp-1.8.0.tgz --name <release_name> --namespace <namespace_name> -f ocats-scp-values-1.8.0.yaml`

   Example: `helm install ocats-scp-1.8.0.tgz --name ocats-scp --namespace scpsvc-f ocats-scp-values-1.8.0.yaml`

   Using Helm 3: `helm3 install <release_name> ocats-scp-1.8.0.tgz -n <namespace_name> -f ocats-scp-values-1.8.0.yaml`

   Example: `helm3 install ocscp-ats ocats-scp-1.8.0.tgz -n scpsvc -f ocats-scp-values-1.8.0.yaml`

> **Note:**
>
> If there are two Helm versions on your system then, specify the version number in the Helm commands. If there is only one Helm version then there is no need to mention the version number.

8. Verify ATS deployment by executing the given command.
   `helm3 status <release_name> -n <namespace_name>`

   The following sample screen checks ATS helm release.

> **Note:**
>
> If ATS is deployed in service mesh environment, the **Ready** field for pods shows 2/2.

Figure 2-22 Checking ATS Helm Release

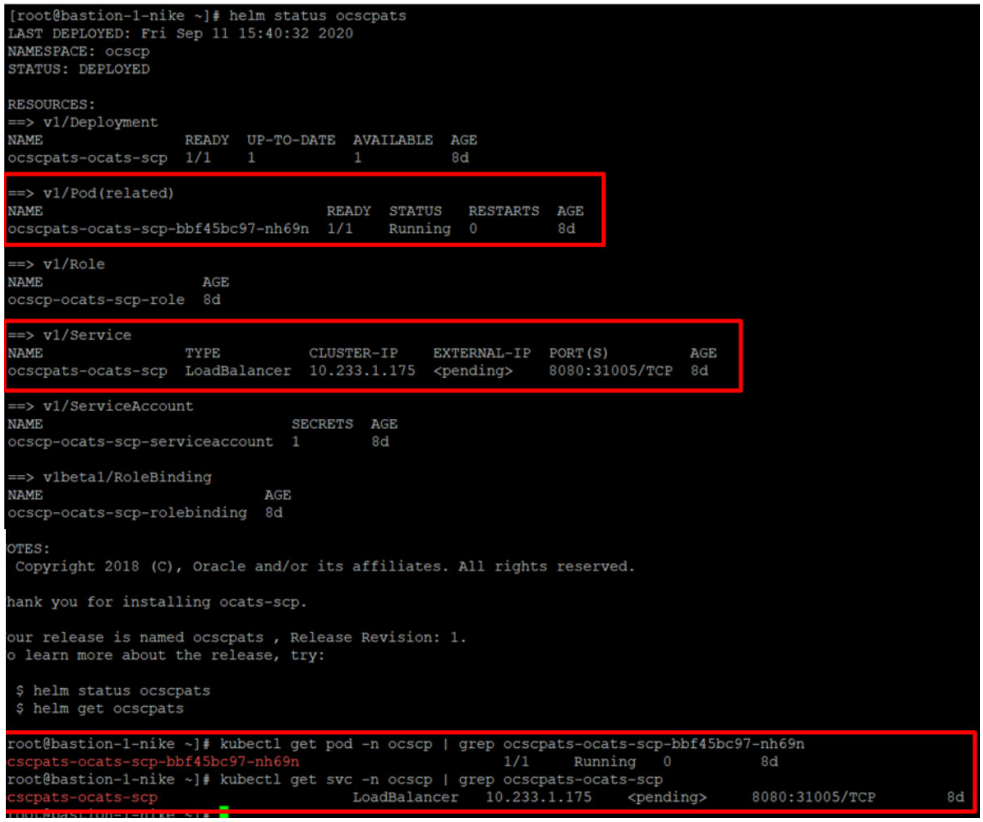
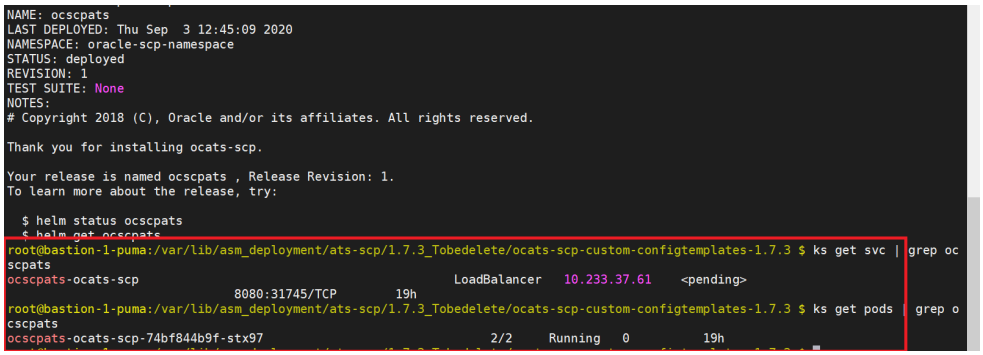


Figure 2-23 Helm Status Image



SEPP ATS Installation Procedure

The SEPP ATS installation procedure covers two steps:

1. Locating and downloading ATS and Simulator Images
2. Deploying ATS and Stub Pod in Kubernetes Cluster as per SEPP

Locating and Downloading ATS Images

1. The ATS Images are available on the OHC server. To download the ATS Images from OHC:
   a. Go to the URL, docs.oracle.com
   b. Navigate to **Industries** > **Communications** > **Cloud Native Core**
   c. Click the **Automated Testing Suite (ATS) Images** link to download the zip file.
   d. Unzip the **Images** folder to access all the ATS Images.
2. The `ocats-sepp` directory has the following files:
   - `ocats-sepp-pkg-1.4.0.0.0.tgz`
   - `ocats-sepp-pkg-1.4.0.0.0-readme.txt`
   - `ocats-sepp-custom-configtemplates-1.4.0.0.0.zip`
   - `ocats-sepp-custom-configtemplates-1.4.0.0.0-readme.txt`

   > **Note:**
   > The `ocats-sepp-pkg-1.4.0.0.0-readme.txt` file contains all the information required for the package.

3. The `ocats-sepp-pkg-1.4.0.0.0.tgz` file has the following images and charts packaged as tar files:

```
ocats-sepp-pkg-1.4.0.0.0.tgz
|
|_ _ _ocats-sepp-pkg-1.4.0.0.tgz
| |_ _ _ _ _ _ ocats-sepp-1.4.0.tgz (Helm Charts)
| |_ _ _ _ _ _ ocats-sepp-image-1.4.tar (Docker Images)
| |_ _ _ _ _ _ Readme.txt
```

4. List of contents in `ocats-sepp-custom-configtemplates-1.4.0.0.0.zip` : `ocats-sepp-custom-configtemplates-1.4.0.0.0.zip`

   |_ _ _ _ _ _ `ocats-sepp-custom-serviceaccount-1.4.0.yaml` (Template to create custom service account)

   |_ _ _ _ _ _ `ocats-sepp-values-1.4.0.yaml` (Custom values file for installation)
5. The user can copy the tar file from here and copy in their OCCNE/OCI/Kubernetes cluster where they want to deploy ATS.

Deploying ATS in Kubernetes Cluster

The steps to deploy ATS in Kubernetes Cluster are as follows:

1. Execute the following command to extract tar file content:
   `tar -xvf ocats-sepp-pkg-1.4.0.0.tgz`

The output of this command is:

- `ocats-sepp-1.4.0.tgz`
- `ocats-sepp-images-1.4.0.tar`
- `Readme.txt`

`ocats-sepp-images-1.4.0.tar` contains ATS Image (`ocats-sepp:1.4.0`) and stub image (`seppstub:1.4.0`).

2. Load images and push to the registry.
3. Create Kubernetes secret with certificates for ats client and stub server. Execute following command to create secret:

```
kubectl create secret generic {secret-name} --from-
file={private_key_file}
 --from-file={trust store password file) --from-file={key store
pasword fie}
 --from-file={private_certificate} --from-file={ca root
certificate} -n {namespace}
```

Example:

```
kubectl create secret generic ocsepp-secret --from-
file=rsa_private_key_pkcs1.pem
 --from-file=trust.txt --from-file=key.txt --from-file=ocsepp.cer --
from-file=caroot.cer -n default
```

> **Note:**
>
> Subject Alternative Name in certificate must be `{ats-helm-release-name}-stubserver.{ats-namespace}` and signing CA must be part of trusted CA of deployed SEPP.

4. Unzip "`ocats-sepp-custom-configtemplates-1.4.0.0.0.zip`". The output of this command is:
   - `ocats-sepp-values-1.4.0.yaml`
   - `ocats-sepp-custom-serviceaccount-1.4.0.yaml`
5. Update image name and tag in `ocats-sepp-values-1.4.0.yaml` file as required.
6. Update Kubernetes secret and certificates details.
7. Execute the below command to deploy ATS :

```
helm install ocats-sepp-1.4.0.tgz --name <release_name>
--namespace <namespace_name> -f ocats-sepp-values-1.4.0.yaml
```

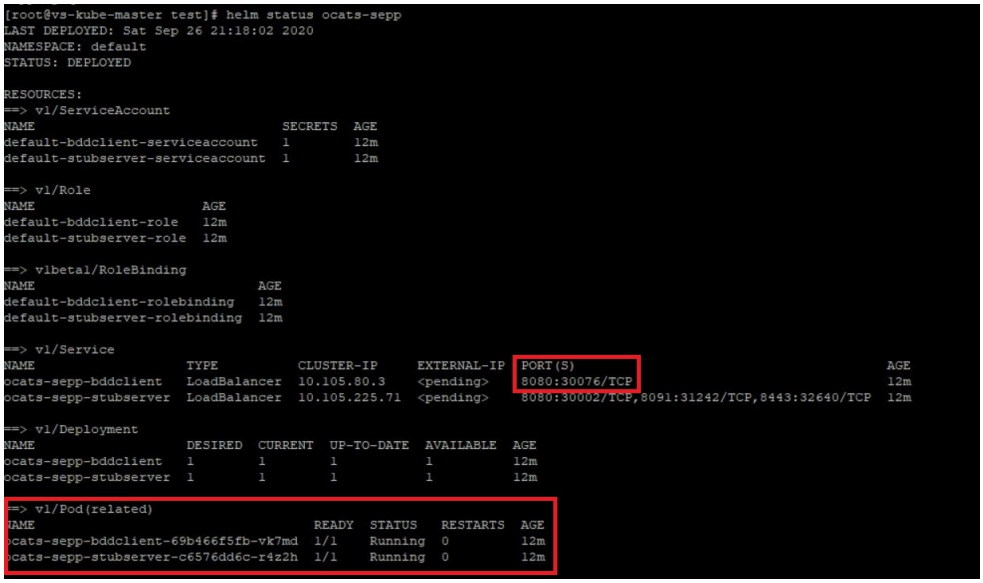
Example:

```
helm install ocats-sepp-1.4.0.tgz --name ocats-sepp --namespace
 seppsvc-f ocats-sepp-values-1.4.0.yaml
```

8. Execute the following command to verify the ATS deployment:
   `helm status <release_name>`

   A sample screen showing ATS Helm release is given below:

Figure 2-24 ATS Helm Release



SLF ATS Installation Procedure

The SLF ATS installation procedure covers two steps:

1. Locating and downloading the ATS images.
2. Deploying ATS images.

Locating and Downloading ATS Images

To locate and download the ATS Images:

1. Download the ATS Images from OHC:
   a. Go to the URL, docs.oracle.com
   b. Navigate to **Industries > Communications > Cloud Native Core**
   c. Click the **Automated Testing Suite (ATS) Images** link to download the zip file.
   d. Unzip the **Images** folder to access all the ATS Images.
2. The `ocats_slf` directory has a following files:

```
ocats-udr-slf-pkg-1.8.0.0.0.tgz

      |_ _ _ _ _ _ ocats-udr-slf-1.8.0.tgz (Helm Charts)

     |_ _ _ _ _ _ ocats-udr-slf-images-1.8.0:1.8.0.tar.tgz (Docker
Images)
```

The user can copy the tar file from here to their kubernetes cluster where they want to deploy ATS.

Preparing to Deploy ATS in Kuberbetes Cluster

To deploy ATS in Kubernetes Cluster:

> **Note:**
>
> Deploy ATS and SLF in the same namespace.

1. Execute the following command to extract the tar file content.
   ```
   tar -xvf ocats-udr-slf-pkg-1.8.0.0.0.tgz
   ```
   The output of this command is:

   ```
   ocats-udr-slf-1.8.0.tgz
   ocats-udr-slf-images-1.8.0:1.8.0.tar.tgz
   ```

   The `ocats-udr-slf-images-1.8.0:1.8.0.tar.tgz` file contains ocats-udr-slf-images-1.8.0 (ATS Image).
2. In your cluster, execute the given command to load the ATS image.
   ```
   docker load --input ocats-udr-slf-images-1.8.0:1.8.0.tar.tgz
   ```
3. Execute the following command to tag and push the ATS image to your registry.

   ```
   docker tag ocats-udr-slf-images-1.8.0:1.8.0 <registry>/ocats-udr-slf-images-1.8.0:1.8.0
   docker push <registry>/ocats-udr-slf-images-1.8.0:1.8.0
   ```

   Example:

   ```
   docker tag ocats-udr-slf-images-1.8.0:1.8.0 localhost:5000/ocats-udr-slf-images-1.8.0:1.8.0
   docker push localhost:5000/ocats-udr-slf-images-1.8.0:1.8.0
   ```
4. Execute the following command to untar the helm charts (ocats-udr-slf-1.8.0.tgz) and update the registry name, image name and tag (if required) in the **ocats-udr-slf/values.yaml** file.

   ```
   tar -xvf ocats-udr-slf-1.8.0.tgz
   ```

   The list of content in ocats-slf is:

   ```
   ocats-slf
           ├── Chart.yaml
           ├── destination-rule-ats.yaml
           ├── templates
           │   ├── deployment.yaml
           │   ├── _helpers.tpl
           │   ├── ingress.yaml
   ```

```
│   ├── NOTES.txt
│   ├── serviceaccount.yaml
│   └── service.yaml
└── values.yaml
```

5. ATS supports static port. By default, this feature is not available. To enable this feature:
   - In the **ocats-udr-slf/values.yaml** file under service section, add the **staticNodePortEnabled** parameter as true and **staticNodePort** parameter with valid nodePort value. A sample screen is given below:

Figure 2-25 ocats-udr-slf/values.yaml - service section

```
service:
  customExtension:
    labels: {}
    annotations: {}
  type: LoadBalancer
  port: "8080"
  staticNodePortEnabled: true
  staticNodePort: "31083"
```

Enabling Service Mesh

To enable service mesh, set the **serviceMeshCheck** parameter to 'true'. This parameter is available under service section of the values.yaml file. A snippet of service section in the yaml file is shown below:

Figure 2-26 Service Mesh Check Enabled

```
service:
  customExtension:
    labels: {}
    annotations: {}
  type: LoadBalancer
  port: "8080"
  staticNodePortEnabled: true
  staticNodePort: "31083"
  serviceMeshCheck: true
```

If service mesh is not enabled at the global level for the namespace then, execute the following command to enable service mesh at the namespace level before deploying ATS.

```
kubectl label --overwrite namespace <namespace_name> istio-
injection=enabled
```

Example:

```
kubectl label --overwrite namespace ocudr istio-injection=enabled
```

> **Note:**
>
> Execute this command only if you are planning to deploy ATS on service mesh enabled system.

Deploying ATS Pod in Kubernetes Cluster

You can deploy ATS Pod in Kubernetes cluster using Helm 2 or Helm 3 commands.

Using Helm 2

Execute the following command to deploy ATS.

```
helm install --name <release_name> --namespace <namespace_name> -f
<values-yaml-file> ocats-udr-slf
```

Example: `helm install --name ocats-udr-slf --namespace ocudr -f ocats-udr-slf/values.yaml ocats-udr-slf`

Using Helm 3

Execute the following command to deploy ATS.

```
helm3 install -name <release_name> --namespace <namespace_name> -f
<values-yaml-file> ocats-udr-slf
```

Example: `helm3 install -name ocats-udr-slf --namespace ocudr -f ocats-udr-slf/values.yaml ocats-udr-slf`

To verify ATS deployment, execute the following command:

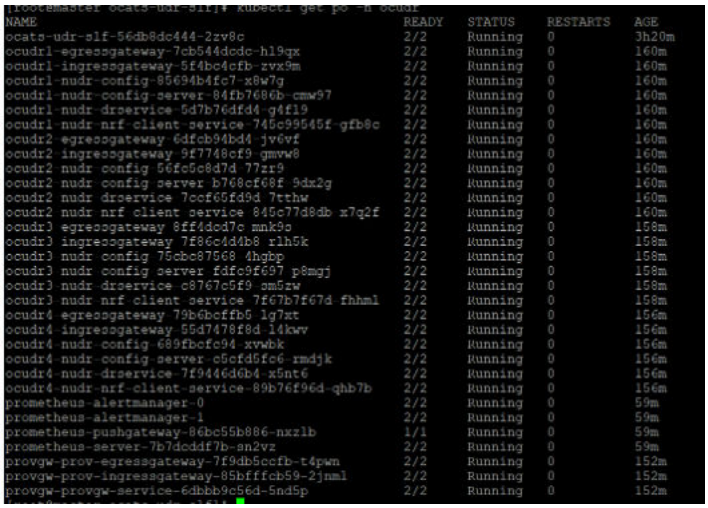
```
helm status <release_name>
```

Figure 2-27 Verifying ATS Deployment

```
[root@master ocats-udr-slf-pkg]# helm get all ocats-udr-slf -n ocudr | grep -A 1 -B 3 STATUS
NAME: ocats-udr-slf
LAST DEPLOYED: Fri Sep 11 14:01:33 2020
NAMESPACE: ocudr
STATUS: deployed
REVISION: 1
[root@master ocats-udr-slf-pkg]#
```

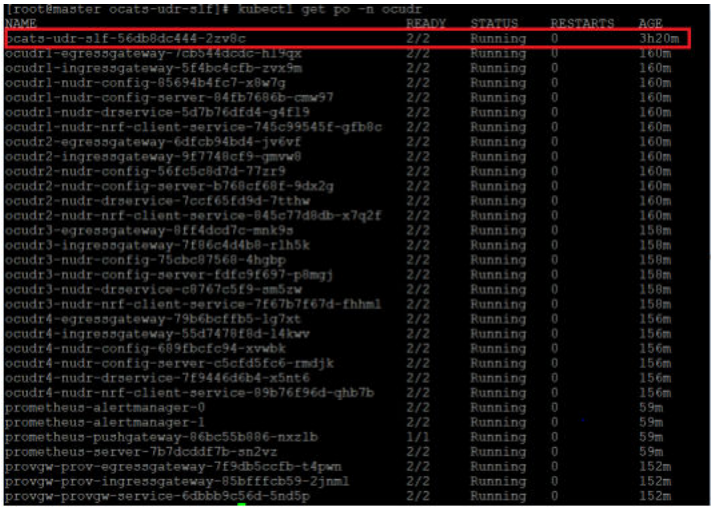
Below is a sample screen showing UDR and ATS installed in the SLF namespace:

Figure 2-28 ATS and SLF Deployed in Same Namespace



If the ATS deployment is done with side car of service mesh, you need to ensure that the ATS shows 2 containers in ready state as "2/2". A sample output of the command is given below:

Figure 2-29 ATS Deployed with Side Car of Service Mesh



Creating a Policy and Destination Rule

Following steps to create a policy are applicable only if a service mesh is enabled at the namespace level:

1. Edit the policy.yaml file as follows:
   - Change the **spec.targets.name** to **ocats-udr-slf svc** name.
   - Change the namespace in which ocats-udr-slf is deployed.

The policy.yaml file snippet is given below:

```
apiVersion: "authentication.istio.io/v1alpha1"
kind: Policy
metadata:
  name: ocats-udr-slf
  namespace: myudr
spec:
  targets:
  - name: ocats-udr-slf
  peers:
  - mtls:
      mode: PERMISSIVE
```

2. Execute the following command to create a policy:
   `kubectl create -f policy.yaml`

   Output: `policy.authentication.istio.io/ocats-udr-slf` is created.

Following steps to create a destination rule are applicable only if a service mesh is enabled at the namespace level:

1. If Service Mesh check is enabled, you need to create a destination rule to fetch the metrics from the Prometheus. This is so because in most of the deployments, Prometheus is kept outside of the service mesh and a destination rule is required to communicate between TLS enabled entity (ATS) and non-TLS entity (Prometheus). To create a destination rule:

   ```
   kubectl apply -f - <<EOF
   apiVersion:networking.istio.io/v1alpha3
   kind:DestinationRule
   metadata:
     name:prometheus-dr
     namespace:myudr
   spec:
     host:oso-prometheus-server.myudr.svc.cluster.local
     trafficPolicy:
       tls:
         mode:DISABLE
   EOF
   ```

   In the above rule,

   - **name** indicates the name of destination rule.
   - **namespace** indicates where the ATS (ocats-udr-slf) is deployed.
   - **host** indicates the hostname of the prometheus server. Change the **spec.host** value to **fqdn** of Prometheus server.

2. Execute the following command to create a destination rule:
   `kubectl create -f destination-rule-ats.yaml`

   Output: `destinationrule.networking.istio.io/ocats-udr-slf-dr` is created.

3
Executing NF Test Cases using ATS

This section describes how to execute NF (NRF ,NSSF, Policy, SCP, SEPP and UDR) Test Cases using ATS.

Executing BSF Test Cases using ATS

This ATS-BSF release is a converged release comprising of scenarios (test cases) from BSF. ATS 1.3.2 is compatible with BSF 1.6.0.

To execute BSF test cases, you need to ensure that following prerequisites are fulfilled.

Prerequisites

- Deploy OCBSF.
- Install Go-STUB in the same namespace where ocbsf is installed.
- ATS Prometheus Metrics validation works only when the installation has a single pod for each microservice in the BSF deployment.
- Users can customize test cases in the custom test case folders (cust_newfeatures, cust_regression and cust_performance). They can add new test cases, remove unwanted test cases and modify existing test cases. It does not impact the original product packaged test cases available in the newfeatures, regression and performance folders. For more details, you can refer to Custom Folder Implementation.
- In the **application-config** configmap, configure the following parameters with the respective values:
  - `primaryNrfApiRoot=http://nf1stub.<namespace_gostubs_are_deployed_in>.svc:8080`
    Example: `primaryNrfApiRoot=http://nf1stub.ocats.svc:8080 #`
  - `secondaryNrfApiRoot=http://nf1stub.ocats.svc:8080 (comment out the secondaryNrfApiRoot)`
  - nrfClientSubscribeTypes=BSF

    > **Note:**
    >
    > To get all configmaps in your namespace execute:
    > `kubectl get configmaps -n <Policy_namespace>`

- **Diameter Log Level Configuration**
  - Set the Log level to Debug in Diam-GW POD:
    `kubectl edit statefulset <diam-gw pod name> -n <namespace>`

    - name: LOGGING_LEVEL_APP

value: DEBUG

- Ensure that the setting for default peer configuration is taken from the config server:
  `kubectl edit statefulset <diam-gw pod name> -n <namespace>`

  - name: USE_CFG_SVC

  value: "true"

- Prometheus server should be installed in cluster.
- Database cluster should be in a running state with all the required tables. You need to ensure that there are no previous entries in database before executing test cases.
- User **MUST NOT** initiate a job in two different pipelines at the same time.
- If Service Mesh check is enabled, then you need to create a destination rule for fetching the metrics from the Prometheus. In most of the deployments, Prometheus is kept outside the service mesh so you need a destination rule to communicate between TLS enabled entity (ATS) and non-TLS entity (Prometheus). You can create a destination rule as follows:

  ```
  kubectl apply -f - <<EOF

  apiVersion:networking.istio.io/v1alpha3
  kind:DestinationRule
  metadata:
    name:prometheus-dr
    namespace:ocats
  spec:
    host:oso-prometheus-server.pcf.svc.cluster.local
    trafficPolicy:
      tls:
        mode:DISABLE
  EOF
  ```

  In the destination rule:

  - **name** indicates the name of destination rule.
  - **namespace** indicates where the ATS is deployed.
  - **host** indicates the hostname of the prometheus server.

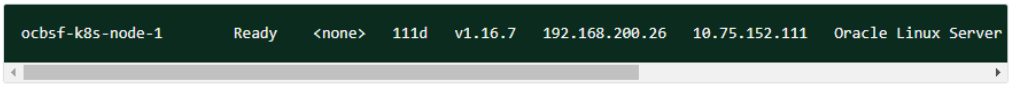
Logging into ATS

Before logging into ATS Jenkins GUI, it is important to get the Worker Node External IP and nodeport of the service, 'ocats-Policy'.

Execute the following command to get the Worker Node External IP:

Example: `kubectl get nodes -owide`

Figure 3-1 Worker Node External IP



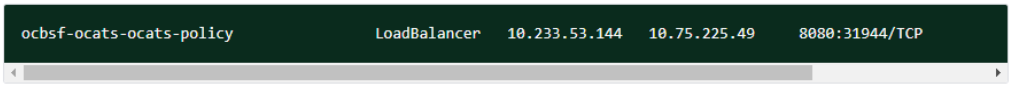
Execute the following command to get the nodeport:

```
kubectl get svc -n <BSF_namespace>
```

Example: `kubectl get svc -n ocbsf`

In the below screenshot, 31944 is the nodeport.

Figure 3-2 BSF Nodeport



To login to Jenkins, open the Web Browser and type the URL: http://<Worker-Node-IP>:<Node-Port-of-ATS>. In the above screen, 31944 is the nodeport. **Example:** http://10.75.225.49:31944

> **Note:**
>
> For more information on ATS deployment in PCF, refer to Policy ATS Installation Procedure.

Executing ATS

To execute ATS:

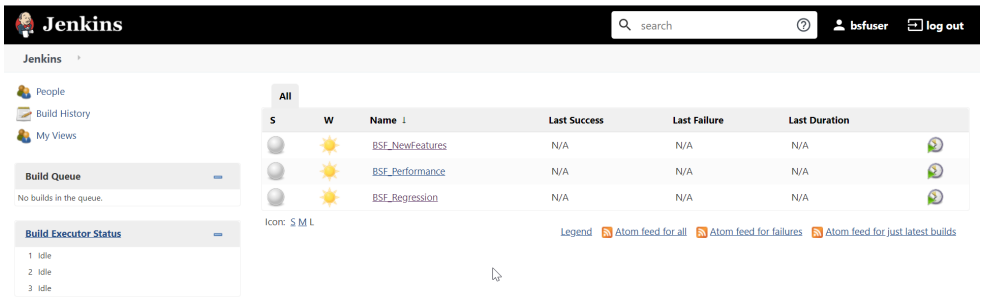
1. Enter the **username** as "bsfuser" and **password** as "bsfpasswd". Click **Sign in**.

   > **Note:**
   >
   > If you want to modify your default login password, refer to Modifying Login Password

   The following screen appears showing BSF pre-configured pipelines:

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

Figure 3-3 Pre-Configured Pipelines



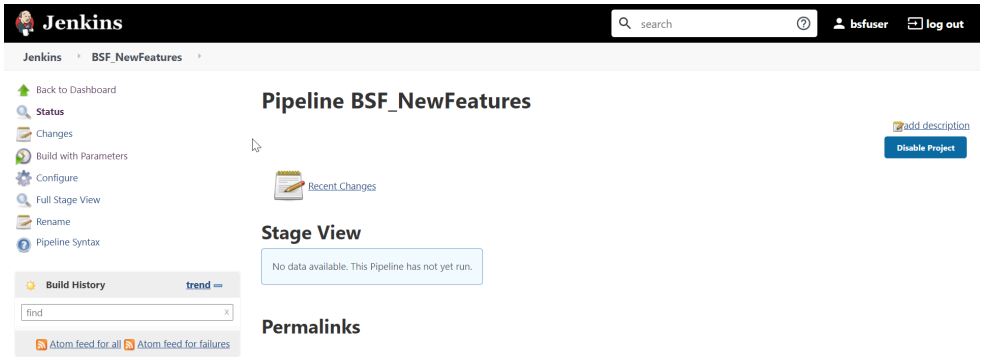
The pre-configured pipelines are explained below:

BSF-New Features Pipeline

This is a pre-configured pipeline where all the BSF new test cases are executed. To configure its parameters, which is a one time activity:

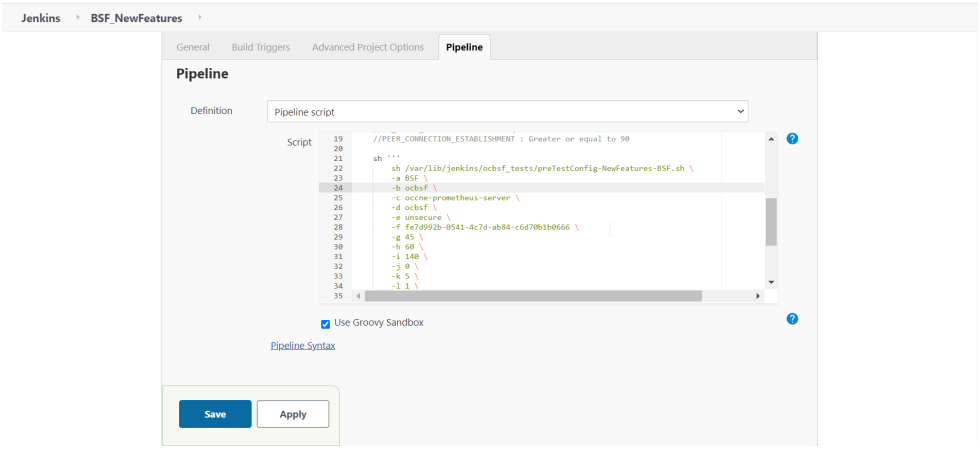
1. Click **BSF-NewFeatures** in the Name column and then, click **Configure** in the left navigation pane as shown below:

Figure 3-4 BSF-NewFeatures Configure



2. The BSF-NewFeatures, **General** tab appears. Make sure that the screen loads completely.
3. Scroll-down to the end. The control moves from **General** tab to the **Pipeline** tab as shown below:

Figure 3-5 BSF - Pipeline Script



In the **Script** area of the Pipeline section, you can change value of the following parameters:

- **b:** Change this parameter to update the namespace where BSF was deployed in your bastion.
- **d:** Change this parameter to update the namespace where your gostubs are deployed in your bastion.
- **e:** Set this parameter as 'unsecure', if you intend to run ATS in TLS disabled mode. Else, set this parameter as 'secure'.
- **g:** Set this parameter to more than 35 secs. The default wait time for the pod is 35 secs. Every TC requires restart of the nrf-client-management pod.
- **h:** Set this parameter to more than 60 secs. The default wait time to add a configurations to the database is 60 secs.
- **i:** Set this parameter to more than 140 secs. The default wait time for Nf_Notification Test Cases is given as 140 secs.
- **k:** Use this parameter to set the waiting time to initialize Test Suite.
- **l:** Use this parameter to set the waiting time to get response from Stub.
- **m:** Use this parameter to set the waiting time after adding BSF Configuration.
- **n:** Use this parameter to set the waiting time for Peer connection establishment.
- **o:** Use this parameter to set the waiting time before sending next message.
- **p:** Use this parameter to set Prometheus Server IP.
- **q:** Use this parameter to set Prometheus Server Port.

> **Note:**
>
> **DO NOT MODIFY ANYTHING OTHER THAN THESE PARAMETER VALUES.**

- Click **Save** after updating the parameters value. The Policy-NewFeatures Pipeline screen appears.

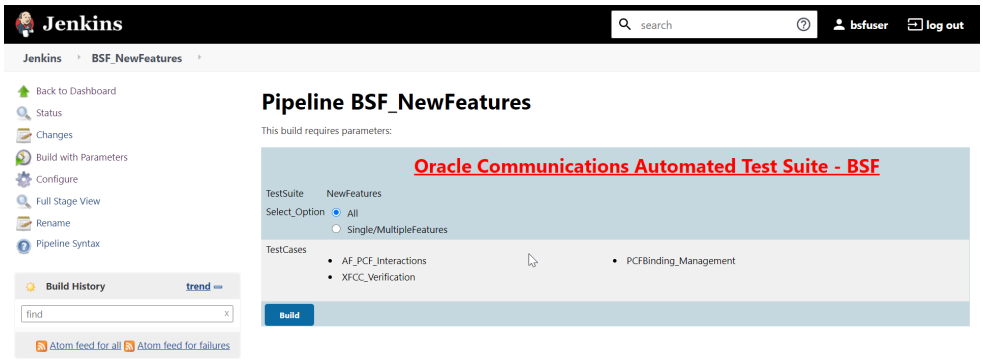
> **Note:**
>
> It is advisable to save the pipeline script in your local machine that you can refer at the time of ATS pod restart.

Executing BSF Test Cases

To execute BSF test cases:

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

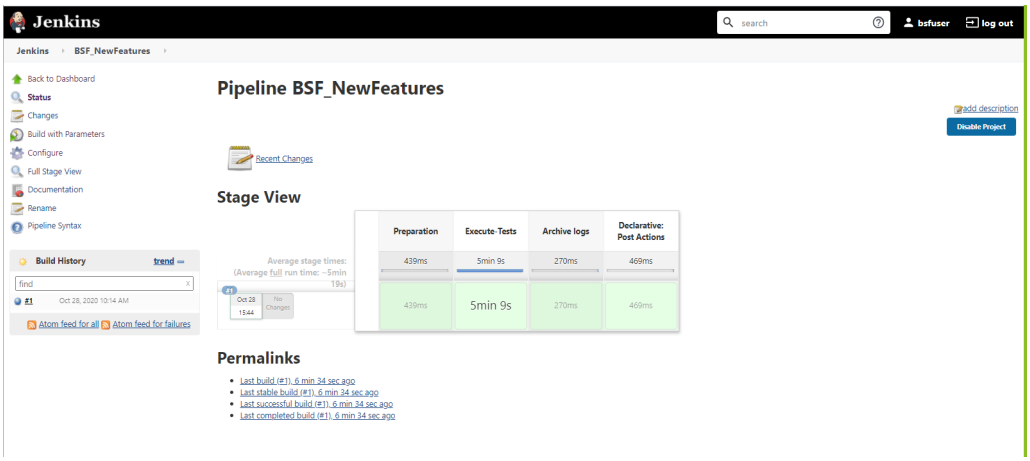
Figure 3-6 BSF - Build with Parameters



Go to **Build** → **Console Output** to view the test result output as shown below:

Figure 3-7 Sample: Test Result Output in Console

Figure 3-8 Sample Output of Build Status



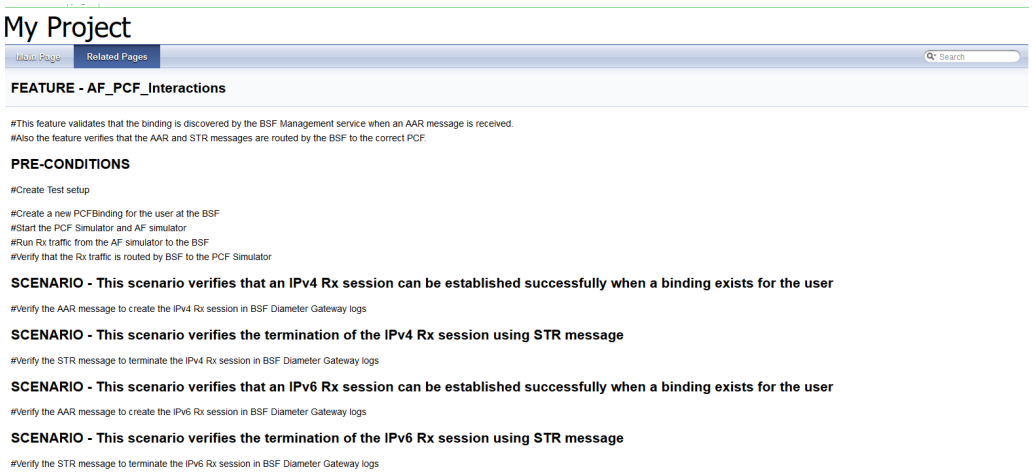
NewFeatures - Documentation

To view Policy functionalities, go to Policy-NewFeatures pipeline and click **Documentation** link in the left navigation pane.

Figure 3-9 BSF-NewFeatures Feature List

You can click any functionality to view its test cases and scenarios of each test case. For example, on click of AF_PCF_Interactions, the following test description appears:

Figure 3-10 Test Cases and Scenarios of Feature-AF_PCF_Interactions



Based on the functionalities covered under Documentation, the **Build Requires Parameters** screen displays test cases. To navigate back to the Pipeline BSF-NewFeatures screen, click **Back to BSF-NewFeatures** link available on top left corner of the screen.

Executing NRF Test Cases using ATS

Prerequisite

To execute NRF Test Cases using NRF ATS 1.8.0, you need to ensure that following prerequisites are fulfilled.

- To execute NF-FQDN-Authentication-Feature test cases, you need to deploy NRF and NRF ATS, both separately with certain changes.
- To execute Geo-Redundancy test cases, you need to deploy two NRF-1.8.0 with replication enabled. These test cases are executed separately as it requires two different NRFs.
- Users can customize test cases in the custom test case folders (cust_newfeatures, cust_regression and cust_performance). They can add new test cases, remove unwanted test cases and modify existing test cases. It does not impact the original product packaged test cases available in the newfeatures, regression and performance folders. For more details, you can refer to Custom Folder Implementation.
- The user should create certificates/keys (public and private) for AccessToken micro-service before deploying NRF.
- Deploy NRF 1.8.0 with default helm configurations using helm charts to execute all cases test except **NF-FQDN-Authentication-Feature**cases.
- All micro-services of NRF should be up and running including Accesstoken micro-service.
- Deploy ATS using helm charts.
- The user **MUST** copy the public keys (RSA and ECDSA) created in the above step to the ATS pod at the **/var/lib/jenkins/ocnrf_tests/public_keys** location.

- Deploy Stub using helm charts.
- For NRF ATS 1.8.0, you need to deploy two stub servers for executing SLF and Forwarding functionality test cases. The service name for both the STUB servers should be **notify-stub-service** and **notify-stub-service02**.
- Ensure Prometheus service is up and running.
- Deploy ATS and Stubs in the same namespace as NRF, as default ATS deployment is with role binding. In addition, deploy test stubs in the same namespace as NRF.
- User **MUST** not initiate a job in two different pipelines at the same time.
- If Service Mesh check is enabled, you need to create a destination rule to fetch the metrics from the Prometheus. This is so because in most of the deployments, Prometheus is kept outside the service mesh and a destination rule is required to communicate between TLS enabled entity (ATS) and non-TLS entity (Prometheus). To create a rule:

  ```
  kubectl apply -f - <<EOF
  apiVersion:networking.istio.io/v1alpha3
  kind:DestinationRule
  metadata:
    name:prometheus-dr
    namespace:ocnrf
  spec:
    host:oso-prometheus-server.ocnrf.svc.cluster.local
    trafficPolicy:
      tls:
        mode:DISABLE
  EOF
  ```

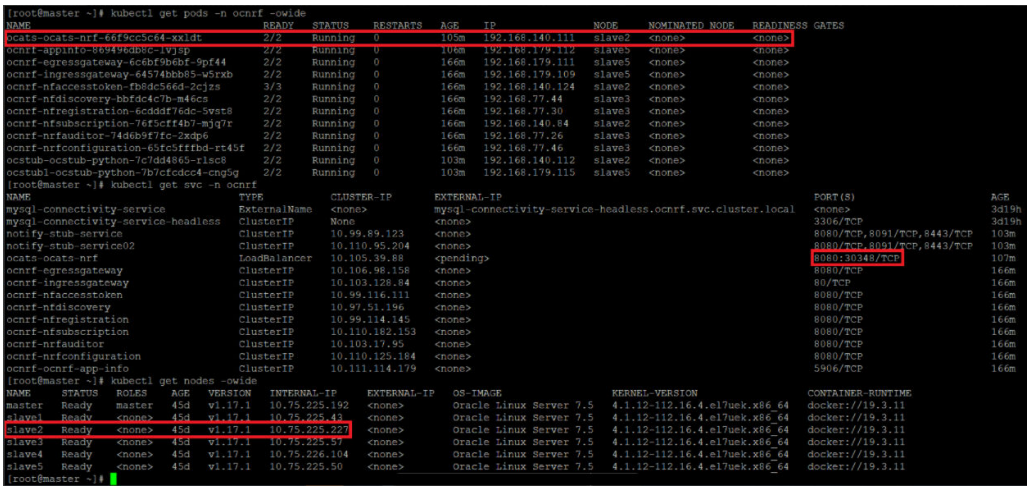
  In the above rule,

  - **name** indicates the name of destination rule.
  - **namespace** indicates where the ATS is deployed.
  - **host** indicates the hostname of the prometheus server.

Logging into ATS

Before logging into ATS, you need to ensure that ATS is deployed successfully using HELM charts. A sample screen is given below:

Figure 3-11 Verifying ATS Pod

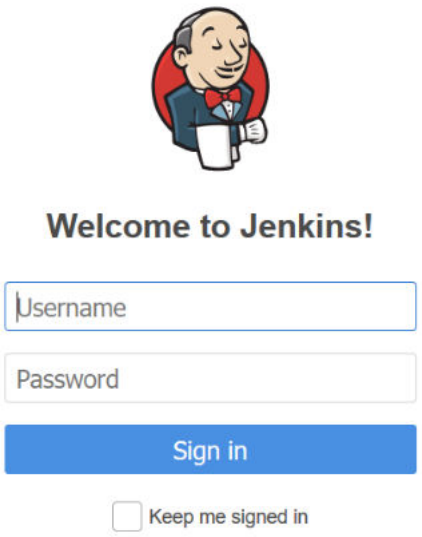


You can use the external IP of the worker node and nodeport of the ATS service as `<Worker-Node-IP>:<Node-Port-of-ATS>`

> **Note:**
>
> In the **Verifying ATS Pod** screen, **slave2** is the node where ATS is deployed, **30348** is the ATS nodeport and **10.75.225.227** is the worker node IP, highlighed in red. For more details on ATS deployment, refer to NRF ATS Installation Procedure.

To login to ATS, open a browser and provide the IP Address and port details as <Worker-Node-IP>:<Node-Port-of-ATS>. As per above screen, it is **10.75.225.227:30348**. The following screen appears:

Figure 3-12 ATS Login

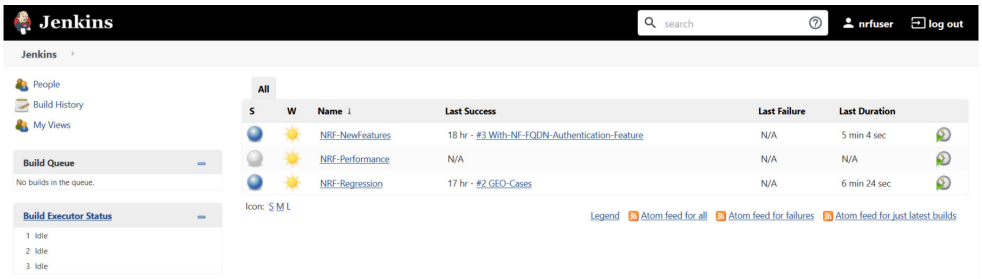


Executing ATS

To execute ATS:

1. Enter the **username** as 'nrfuser' and **password** as 'nrfpasswd'. Click **Sign in**. The following screen appears.

Figure 3-13 NRF Pre-Configured Pipelines



NRF ATS has three pre-configured pipelines.

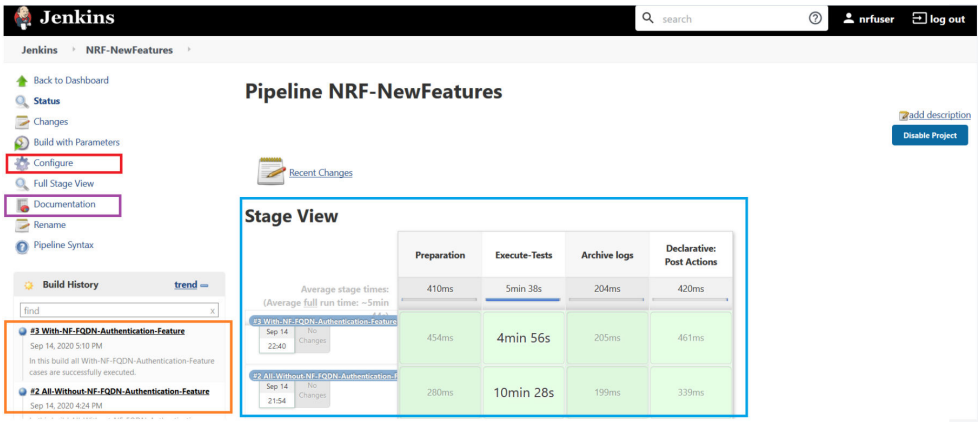
- **NRF-NewFeatures:** This pipeline has all the test cases, which are delivered as part of NRF ATS - 1.8.0
- **NRF-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
- **NRF-Regression:** This pipeline has all the test cases delivered so far in the previous releases.

NRF-NewFeatures Pipeline

After identifying the NRF pipelines, the user needs to do one-time configuration in ATS as per NRF deployment. In this pipeline, all the new testcases related to NRF are executed. To configure its parameters:

1. Click **NRF-NewFeatures** in the Name column. Following screen appears:

Figure 3-14 Configuring NRF-New Features

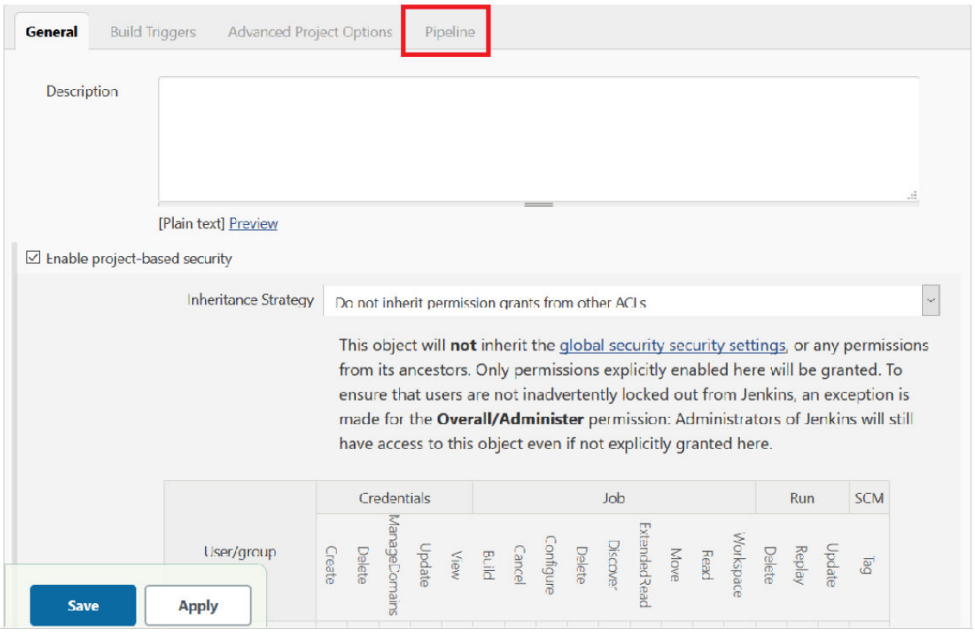


In the above screen:

- Click **Configure** to navigate to the screen where configuration needs to be done.
- Click **Documentation** to view the documented test cases, which are part of this NRF release.
- Click the blue dots inside **Build History** box to view the success console logs of the "Sanity", "All-Without-NF-FQDN-Authentication-Feature" and "With-NF-FQDN-Authentication-Feature" respectively.
- The **Stage View** represents the already executed pipeline for the customer reference.

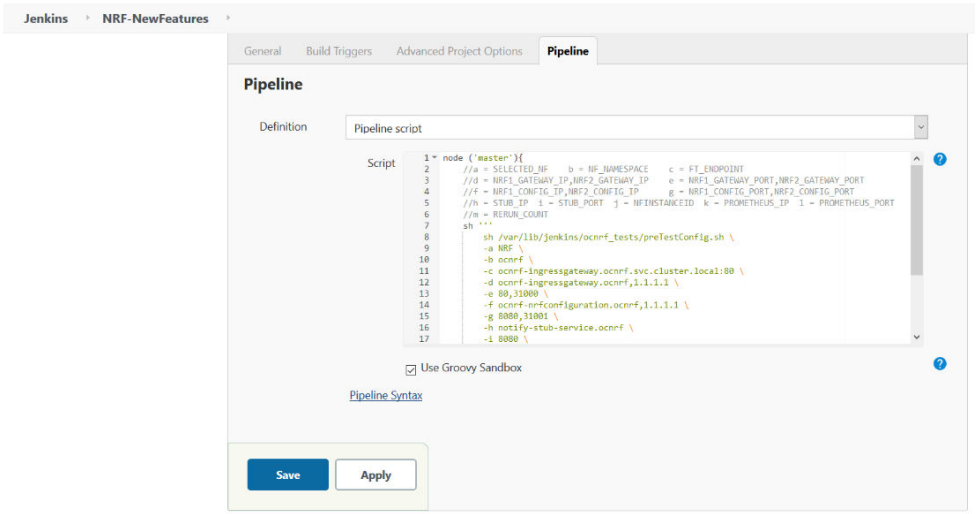
2. Click **Configure**. User **MUST** wait for the page to load completely. Once the page loads completely, click the **Pipeline** tab as shown below:
MAKE SURE THAT THE SCREEN SHOWN BELOW LOADS COMPLETELY BEFORE YOU PERFORM ANY ACTION ON IT. ALSO, DO NOT MODIFY ANY CONFIGURATION OTHER THAN DISCUSSED BELOW.

Figure 3-15 Pipeline Option



3. The **Pipeline** section of the configuration page appears as shown below:

Figure 3-16 Pipeline Section



In the above screen, you can change the values of the '**Pipeline script**'. The content of the pipeline script is as follows:

Figure 3-17 Pipeline Script

```
1  node ('master'){
2      //a = SELECTED_NF    b = NF_NAMESPACE    c = FT_ENDPOINT
3      //d = NRF1_GATEWAY_IP,NRF2_GATEWAY_IP    e = NRF1_GATEWAY_PORT,NRF2_GATEWAY_PORT
4      //f = NRF1_CONFIG_IP,NRF2_CONFIG_IP      g = NRF1_CONFIG_PORT,NRF2_CONFIG_PORT
5      //h = STUB_IP  i = STUB_PORT  j = NFINSTANCEID  k = PROMETHEUS_IP  l = PROMETHEUS_PORT
6      //m = RERUN_COUNT
7      sh '''
8          sh /var/lib/jenkins/ocnrf_tests/preTestConfig.sh \
9          -a NRF \
10         -b ocnrf \
11         -c ocnrf-ingressgateway.ocnrf.svc.cluster.local:80 \
12         -d ocnrf-ingressgateway.ocnrf,1.1.1.1 \
13         -e 80,31000 \
14         -f ocnrf-nrfconfiguration.ocnrf,1.1.1.1 \
15         -g 8080,31001 \
16         -h notify-stub-service.ocnrf \
17         -i 8080 \
18         -j 6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c \
19         -k occne-prometheus-server.occne-infra \
20         -l 80 \
21         -m 0
22     '''
23     load "/var/lib/jenkins/ocnrf_tests/jenkinsData/Jenkinsfile-NewFeatures"
24 }
```

> **Note:**
>
> The User **MUST NOT** change any other value apart from **line number 9** to **line 21**.

You can change the parameter values from **"a"** - to - **"m"** as per user requirement. The parameter details are available as comments from line number **2** - to - **6**.

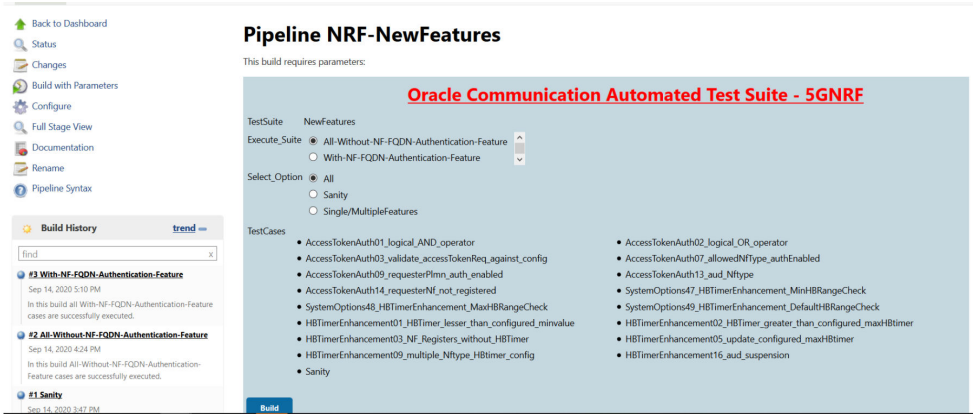
a: Name of the NF to be tested in capital (NRF).
b: Namespace in which the NRF is deployed.
c: endPointIP:endPointPort value used while deploying NRF with the help of helm chart.
d: Comma separated values of NRF1 and NRF2 ingress gateway service (ocnrf-ingressgateway.ocnrf,1.1.1.1). It is also known as as cluster_domain. A dummy value
of NRF2 ingress gateway (1.1.1.1) is provided for the reference.
e: Comma separated values of NRF1 and NRF2 port of ingressgateway service (80,31000).
A dummy value of NRF2 ingress gateway port (31000) is provided for the reference.
f: Comma separated values of NRF1 and NRF2 configuration service (ocnrf-nrfconfiguration.ocnrf,1.1.1.1). It is also known as as cluster_domain.
A dummy value of NRF2 configuration service (1.1.1.1) is provided for the reference.
g: Comma separated values of NRF1 and NRF2 port of configuration service (8080,31001).
A dummy value of NRF2 configuration microservice port (31001) is provided for the
reference.
h: Name_of_stub_service.namespace (notify-stub-service.ocnrf).

```
i: Port of stub service (8080).
j: NRF_Instance ID (6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c).
k: Name_of_Prometheus_service.namespace (occne-prometheus-
server.occne-infra).
l: Port of Prometheus service (80).
m: Number of times the re-run of failed case is allowed (default as
0).
```

> **Note:**
>
> You need not to change any value if
>
> - OCCNE cluster is used
> - NRF, ATS and Stub are deployed in the ocnrf namespace
>
> If any GEO-Redundancy case is executed, you have to provide pipeline script values for NRF-2 in d, e, f and g options as per deployment.

4. Click **Save** after making neccesary changes. The NRF-NewFeatures screen appears. Click **Build with Parameters**. Following screen appears:

Figure 3-18 Pipeline NRF-NewFeatures



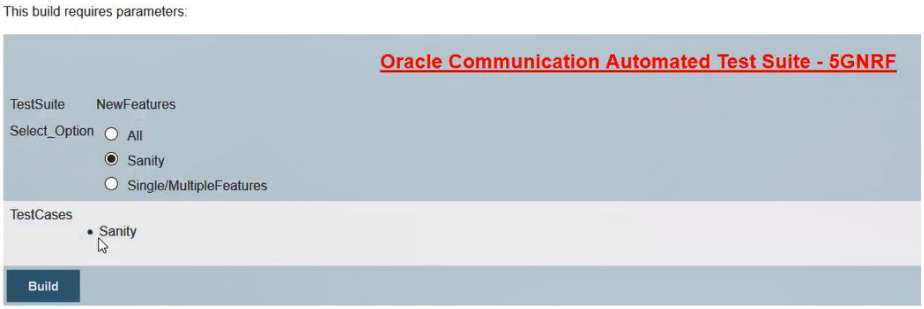
In the above screen, you have **Execute_Suite** options to execute NRF test cases either:

- **All-Without-NF-FQDN-Authentication-Feature**: This is the default option. It executes all the test cases except **NF-FQDN-Authentication-Feature**.
- **With-NF-FQDN-Authentication-Feature**: It executes all **NF-FQDN-Authentication-Feature** test cases.

In the above screen, there are three **Select_Option**(s), which are:

- **All:** This is the default option. It executes all the NRF test cases. User just need to scroll down and click **Build** to execute all the test cases.
- **Sanity:** It is recommended to execute Sanity before executing any test case. This helps to ensure that all the deployments are done properly. When you select Sanity, the following screen appears:

Figure 3-19 Build Requires Parameters - Sanity



Click **Build** to execute all the sanity test cases.

> **Note:**
>
> Sanity option is not available when Execute_Suite is set to **With-NF-FQDN-Authentication-Feature**.

- **Single/MultipleFeatures:** This option allows you to select any number of test cases that you want to execute from the list of total test cases available for execution. After selecting the test cases, scroll-down and click **Build**. The selected NRF test cases are executed.

The NRF testcases are divided into following NRF Service operations:

- **NRF Sanity** - This feature file contains all the basic sanity test cases of NRF ATS to validate whether the deployment is correct or not. It is advisable to execute these test cases before starting a complete suite.
- **Configuration** - These feature files are listed with a prefix as **"SystemOptions"**.
- **Registration** - These feature files are listed with a prefix as **"HBTimerEnhancement01"**.
- **AccessToken** - These feature files are listed with a prefix as **"AccessTokenAuth"**.
- **NF-FQDN-Authentication** - These feature files are listed with a prefix as **"NfAuthentication"**.

The following screen shows successful execution of **Sanity**, **All-Without-NF-FQDN-Authentication-Feature** and **With-NF-FQDN-Authentication-Feature** test cases.

Figure 3-20 Sample Screen: NRF-ATS Full Execution

The following screens show the results for **Sanity**, **All-Without-NF-FQDN-Authentication-Feature** and **With-NF-FQDN-Authentication-Feature** test cases in the same order as they are executed.

Figure 3-21 Test Cases Result - Sanity

```
1 feature passed, 0 failed, 0 skipped
11 scenarios passed, 0 failed, 0 skipped
209 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m58.426s
[Pipeline] sh
+ cd /var/lib/jenkins/ocnrf_tests
++ cat /var/lib/jenkins/ocnrf_tests/environ.sh
++ grep RERUN
++ cut -d= -f2
++ cut '-d;' -f1
+ rerun=0
+ sh re-run.sh 0
0
Success
```

Figure 3-22 Test Cases Result - All-Without-NF-FQDN-Authentication-Feature

```
17 features passed, 0 failed, 0 skipped
46 scenarios passed, 0 failed, 0 skipped
746 steps passed, 0 failed, 0 skipped, 0 undefined
Took 7m10.187s
[Pipeline] sh
+ cd /var/lib/jenkins/ocnrf_tests
++ cat /var/lib/jenkins/ocnrf_tests/environ.sh
++ grep RERUN
++ cut -d= -f2
++ cut '-d;' -f1
+ rerun=0
+ sh re-run.sh 0
0
Success
```

Figure 3-23 Test Cases Result - All-With-NF-FQDN-Authentication-Feature

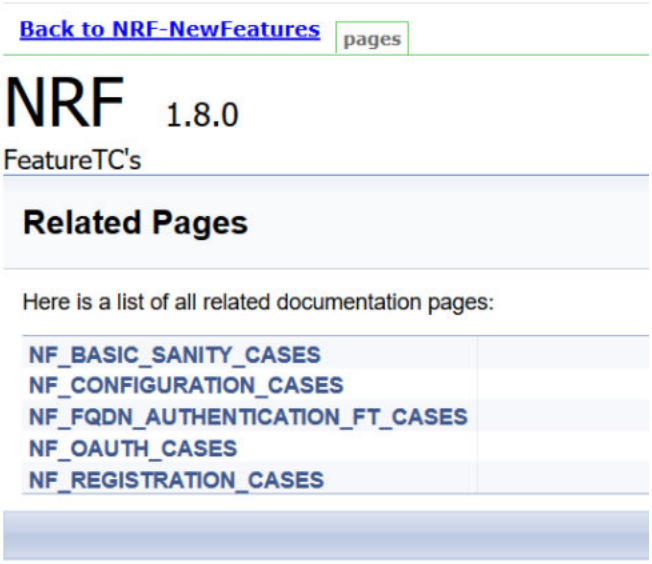
```
5 features passed, 0 failed, 0 skipped
20 scenarios passed, 0 failed, 0 skipped
340 steps passed, 0 failed, 0 skipped, 0 undefined
Took 3m9.773s
[Pipeline] sh
+ cd /var/lib/jenkins/ocnrf_tests
++ cat /var/lib/jenkins/ocnrf_tests/environ.sh
++ grep RERUN
++ cut -d= -f2
++ cut '-d;' -f1
+ rerun=0
+ sh re-run.sh 0
0
Success
```

NRF-NewFeatures Documentation

To view NRF test cases, go to NRF-NewFeatures pipeline and click **Documentation** link in the left navigation pane. It shows all the test cases provided as part of NRF

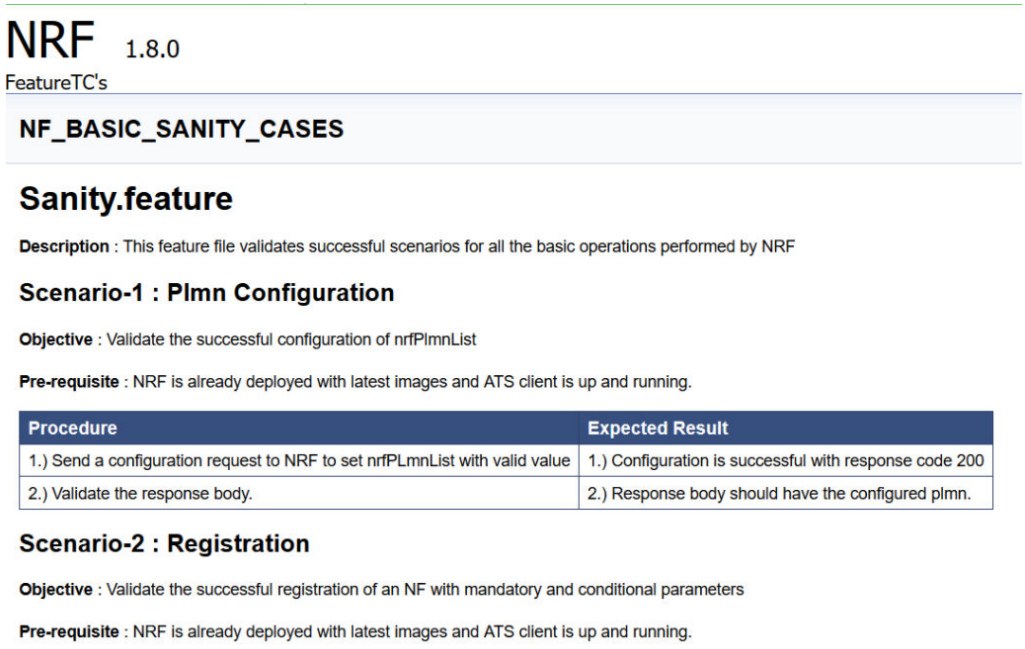
ATS -1.8.0 along with sanity cases. The following screen shows all the documentation features:

Figure 3-24 NRF-NewFeatures Documentation



Click any functionality to view its test cases and scenarios of each test case. A sample screen is as follows:

Figure 3-25 Sample Feature: NF_BASIC_SANITY_CASES



Based on the functionalities covered under Documentation, the **Build Requires Parameters** screen displays test cases. To navigate back to the Pipeline NRF-

NewFeatures screen, click **Back to NRF-NewFeatures** link available on top left corner of the screen.

NRF-Regression Pipeline

This pre-configured pipeline contains all the test cases that are delivered till NRF ATS 1.7.0. However, some test cases are updated as per new implementation of NRF.

The configuration method and parameters are same as the **NewFeatures** pipeline. Only difference in this pipeline is that it does not have **Sanity** option. Thus to configure this pipeline, you have to provide NRF2 details.

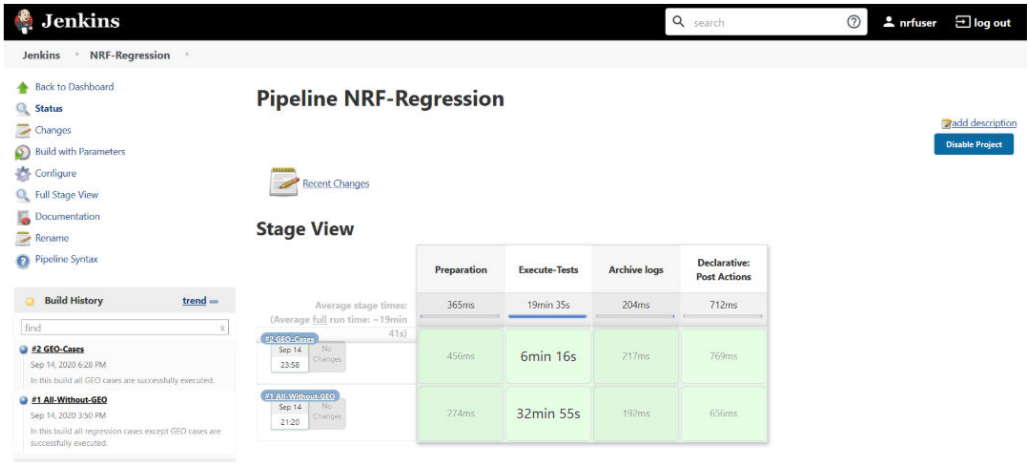
From this release onwards, GEO will be part of Regression, so please correct it. NRF2 details are required to be provided while configuring the Regression pipeline.

The NRF-Regression test cases are divided into following service operations:

- **AccessToken** - These feature files are listed with a prefix as **"oAuth"**.
- **Configuration** - These feature files are listed with a prefix as **"Config"**.
- **Discovery** - These feature files are listed with a prefix as **"Disc"**.
- **NRF Forwarding** - These feature files are listed with a prefix as **"Forwarding"**.
- **NRF Functional** - These feature files are listed with a prefix as **"Feat"**.
- **Registration** - These feature files are listed with a prefix as **"Reg"** and **"Upd"**. These are related to update operation of registered profiles.
- **NRF SLF** - These feature files are listed with a prefix as **"SLF"**.
- **Subscription** - These feature files are listed with a prefix as **"Subs"**.
- **Geo Redundancy** - These feature files are listed with a prefix as **"Geo"**.

> **Note:**
>
> You need not to change any value if any GEO-Redundancy case is not executed. If any GEO-Redundancy case is executed, you have to provide pipeline script values for NRF-2 in d, e, f and g options as per deployment.

Figure 3-26 NRF-Regression



The following screen shows full successful execution as part of ATS image.

Figure 3-27 NRF-Regression - All-Without-GEO

```
237 features passed, 0 failed, 0 skipped
557 scenarios passed, 0 failed, 0 skipped
8991 steps passed, 0 failed, 0 skipped, 0 undefined
Took 24m50.625s
[Pipeline] sh
+ cd /var/lib/jenkins/ocnrf_tests
++ cat /var/lib/jenkins/ocnrf_tests/environ.sh
++ cut '-d;' -f1
++ grep RERUN
++ cut -d= -f2
+ rerun=0
+ sh re-run.sh 0
0
Success
```

Figure 3-28 NRF-Regression - GEO Cases

```
6 features passed, 0 failed, 0 skipped
15 scenarios passed, 0 failed, 0 skipped
418 steps passed, 0 failed, 0 skipped, 0 undefined
Took 3m39.979s
[Pipeline] sh
+ cd /var/lib/jenkins/ocnrf_tests
++ cat /var/lib/jenkins/ocnrf_tests/environ.sh
++ grep RERUN
++ cut -d= -f2
++ cut '-d;' -f1
+ rerun=0
+ sh re-run.sh 0
0
Success
```

NRF-Regression Documentation

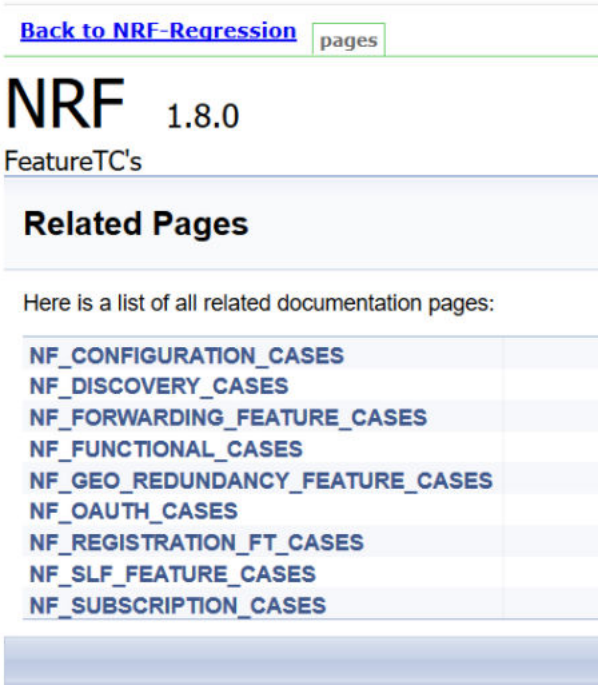
Click **Documentation** in the left navigation pane of the NRF-Regression pipeline to view all the test cases provided till NRF ATS 1.7.0.

The NRF test cases are divided into multiple groups based on following functionalities:

- **NF_CONFIGURATION_CASES** - Lists the cases related to NRF configuration.
- **NF_DISCOVERY_CASES** - Lists all the discovery microservice related cases.
- **NF_FORWARDING_FEATURE_CASES** - Lists all the forwarding related cases.
- **NF_FUNCTIONAL_CASES** - Lists all the functional cases.
- **NF_GEO_REDUNDANCY_FEATURE_CASES** - Lists all the Geo-Redundancy related cases.
- **NF_OAUTH_CASES** - Lists all the accesstoken related cases.
- **NF_REGISTRATION_CASES** - Lists all the registration related cases.
- **NF_SLF_FEATURE_CASES** - Lists all the SLF related cases.
- **NF_SUBSCRIPTION_CASES** - Lists all subscription related cases.

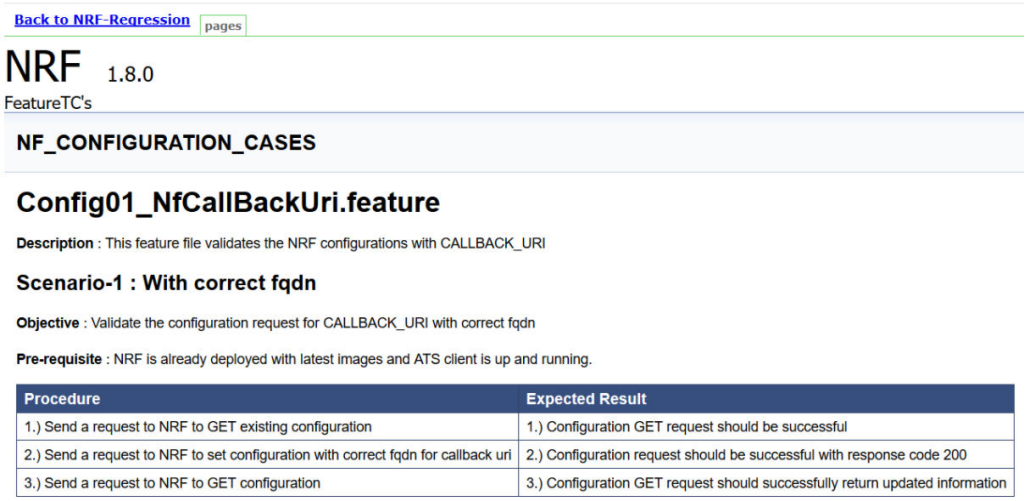
Following screen appears:

Figure 3-29 NRF-Regression Documentation



A sample screen showing documentation of Regression pipeline for NRF ATS - 1.8.0 is given below:

Figure 3-30 Sample Screen: NRF-Regression Documentation



Back to NRF-Regression pages

NRF 1.8.0

FeatureTC's

NF_CONFIGURATION_CASES

Config01_NfCallBackUri.feature

Description : This feature file validates the NRF configurations with CALLBACK_URI

Scenario-1 : With correct fqdn

Objective : Validate the configuration request for CALLBACK_URI with correct fqdn

Pre-requisite : NRF is already deployed with latest images and ATS client is up and running.

Procedure	Expected Result
1.) Send a request to NRF to GET existing configuration	1.) Configuration GET request should be successful
2.) Send a request to NRF to set configuration with correct fqdn for callback uri	2.) Configuration request should be successful with response code 200
3.) Send a request to NRF to GET configuration	3.) Configuration GET request should successfully return updated information

Executing NSSF Test Cases using ATS

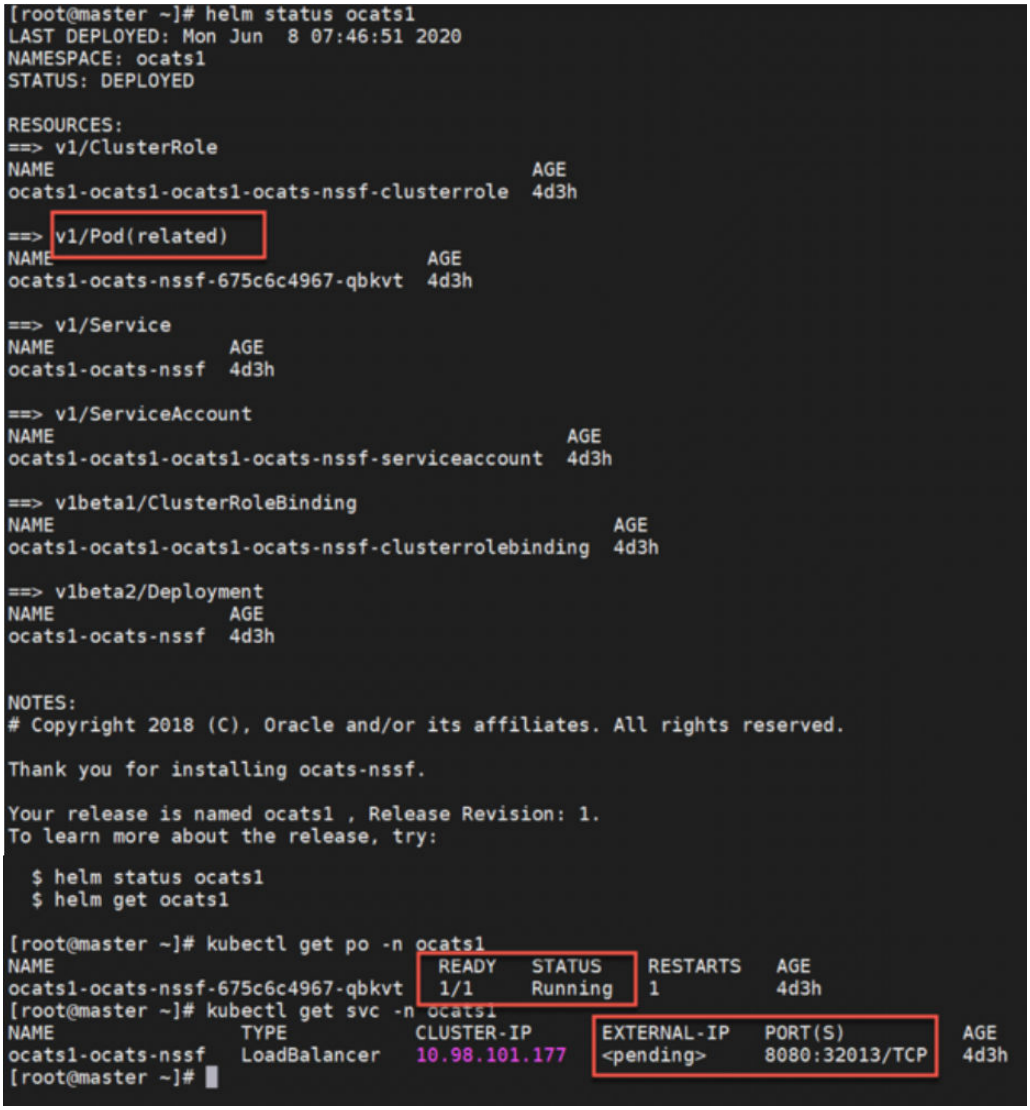
To execute NSSF Test Cases using NRF ATS 1.4, you need to ensure that following prerequisites are fulfilled.

- Before deploying NSSF, the user must create certificates/keys (public and private) for AccessToken microservice. The public keys (RSA and ECDSA) must be copied to the ATS pod at **/var/lib/jenkins/ocnssf_tests/public_keys** location.
- User must deploy NSSF 1.4 with default helm configurations using helm charts.
- All NSSF micro-services should be up and running including AccessToken microservice.

Logging into ATS

Before logging into ATS, you need to ensure that ATS is deployed successfully using HELM charts. A sample screen is given below:

Figure 3-31 Verifying ATS Deployment



There are two ways to login to ATS Jenkins GUI.

- When an external load balancer (metalLB in case of OCCNE) is available and an external IP is provided to the ATS service, the user can login to ATS GUI using <External-IP>:8080.
- When an external IP is not provided to the ATS service, the user can open the browser and provide the external IP of the worker node and nodeport of the ATS service to login to ATS GUI.
  `<Worker-Node-IP>:<Node-Port-of-ATS>`

> **Note:**
>
> In the **Verifying ATS Deployment** screen, ATS nodeport is highlighted in red as 32013. For more details on ATS deployment, refer to **NSSF ATS Installation Procedure.**

Open a browser and provide IP and port details as <Worker-Node-IP>:<NodePort-of-ATS> (As per the above example: 10.98.101.171:32013). The ATS login screen appears.

Executing ATS

To execute ATS:

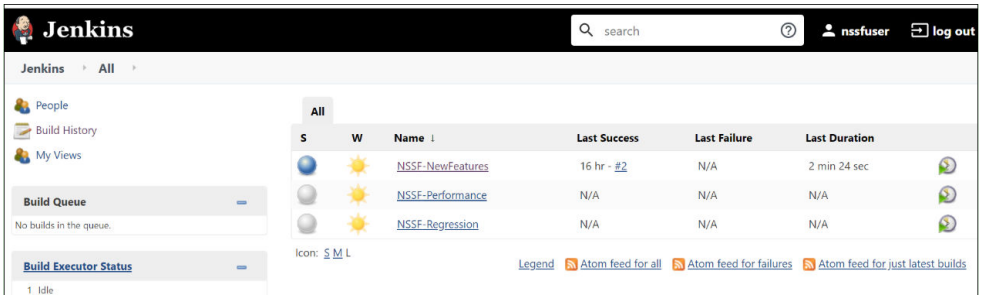
1. Enter the username as 'nssfuser' and password as 'nssfpasswd'. Click **Sign in**.

> **Note:**
>
> If you want to modify your default login password, refer to Modifying Login Password

The following screen appears showing pre-configured pipelines for NSSF individually (3 Pipelines).

- **NSSF-New-Features:** This pipeline has all the test cases that are delivered as part of NSSF ATS - 1.4.
- **NSSF-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
- **NSSF-Regression:** This pipleine has all the test cases of previous releases. As this is the first release of NSSF-ATS, this pipeline does not show any previous release test cases.

Figure 3-32 Pre-Configured Pipelines

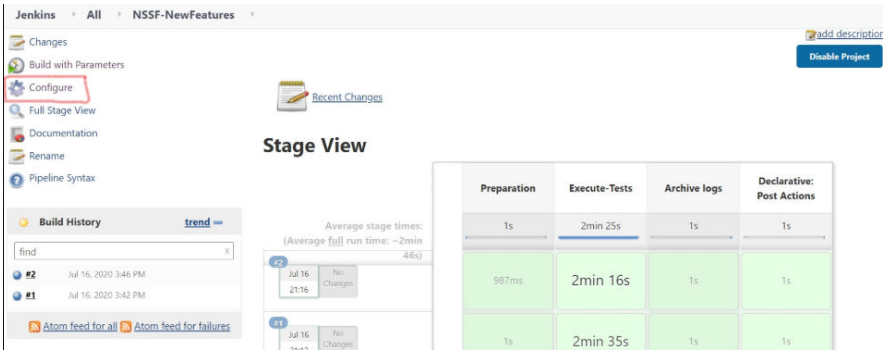


Each one of this pipeline is explained below:

- **NSSF-NewFeatures Pipeline:** After identifying the NSSF pipelines, the user needs to do one-time configuration in ATS as per their SUT deployment. In this pipeline, all the new testcases related to NSSF are executed. To configure its parameters:

a. Click **NSSF-NewFeatures** in the Name column. The following screen appears:

Figure 3-33 NSSF-NewFeatures Pipeline

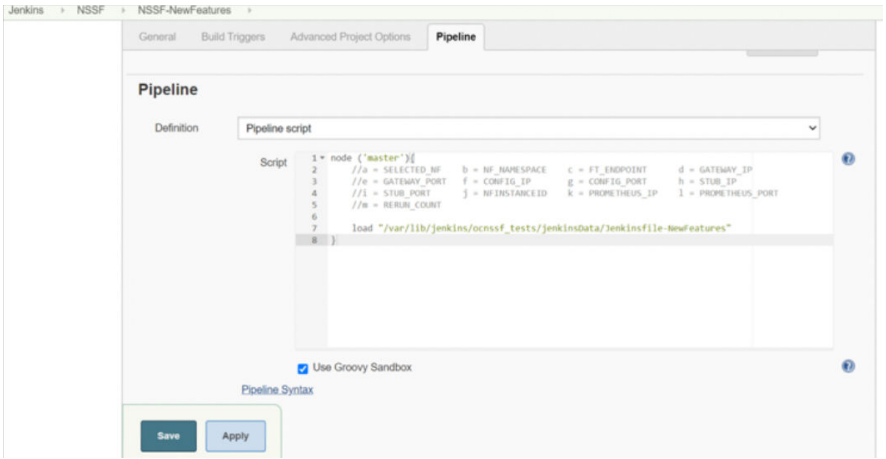


In the above screen:

- Click **Configure** to navigate to a screen where configuration needs to be done.
- Click **Documentation** to view the documented test cases.
- Click blue dots inside **Build History** box to view the success console logs of the "All" and "Sanity" respectively.
- The **Stage View** represents already executed pipeline for the customer reference.

b. Click **Configure**. Users MUST wait for the page to load completely. Once the page loads completely, click the **Pipeline** tab to reach the Pipeline configuration as shown below:
MAKE SURE THAT THE SCREEN SHOWN ABOVE LOADS COMPLETELY BEFORE YOU PERFORM ANY ACTION ON IT. ALSO, DO NOT MODIFY ANY CONFIGURATION OTHER THAN DISCUSSED BELOW.

Figure 3-34 NSSF Configure



c. In the above screen, the values of the 'Pipeline script' needs to be changed. The content of the pipeline script is as follows:

```
node ('master'){
    //a = SELECTED_NF    b = NF_NAMESPACE    c =
FT_ENDPOINT      d = GATEWAY_IP
    //e = GATEWAY_PORT   f = CONFIG_IP       g =
CONFIG_PORT      h = STUB_IP
    //i = STUB_PORT      j = NFINSTANCEID    k =
PROMETHEUS_IP    l = PROMETHEUS_PORT
    //m = RERUN_COUNT
    sh '''
        sh /var/lib/jenkins/ocnssf_tests/preTestConfig.sh \
        -a NSSF \
        -b ocnssf \
        -c ocnssf-ingressgateway.ocnssf.svc.cluster.local:80
\
        -d ocnssf-ingressgateway.ocnssf \
        -e 80 \
        -f ocnssf-nssfconfiguration.ocnssf \
        -g 8080 \
        -h notify-stub-service.ocnssf \
        -i 8080 \
        -j 6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c \
        -k occne-prometheus-server.occne-infra \
        -l 80 \
        -m 2
    '''
    load "/var/lib/jenkins/ocnssf_tests/jenkinsData/
Jenkinsfile-NewFeatures"
}
```

> **Note:**
>
> The User MUST NOT change any other value apart from line number 8 to line 20.

You can change only those parameters that are marked as "a" to "m" as per your requirement.

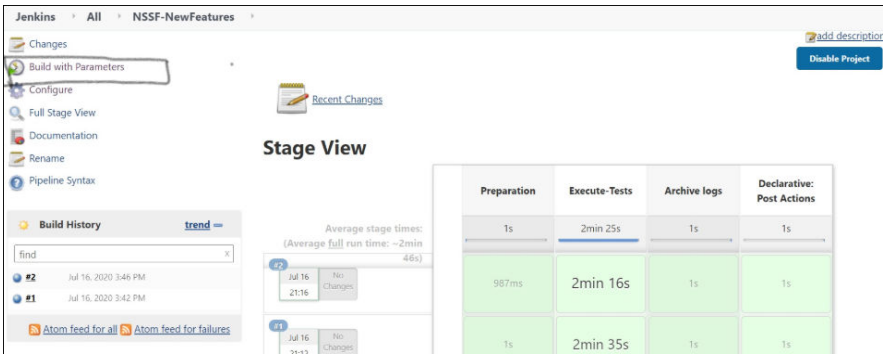
- a - Name of the NF to be tested in capital (NSSF).
- b - Namespace in which the NSSF is deployed
- c - endPointIP:endPointPort value used while deploying the NSSF using the helm chart
- d - Name_of_NSSF_ingressgateway_service.namespace (ocnssf-nssfconfiguration.ocnssf) - this is also known as as cluster_domain.
- e - Port of ingressgateway service (80)
- f - Name_of_NSSF_configuration_service.namespace (ocnssf-nssfconfiguration.ocnssf)
- g - Port of configuration service (8080)

- h - Name_of_stub_service.namespace (notify-stub-service.ocnssf)
- i - Port of stub service (8080)
- j - NSSF_Instance ID (6faf1bbc-6e4a-4454-a507-a14ef8e1bc5c)
- k - Name_of_Prometheus_service.namespace (occne-prometheus-server.occne-infra)
- l - Port of Prometheus service (80)
- m - Number of times the re-run of failed case is allowed (default as 2).

> **Note:**
>
> You do not have to change any value if OCCNE cluster is used and NSSF, ATS and STUB are deployed in ocnssf namespace.

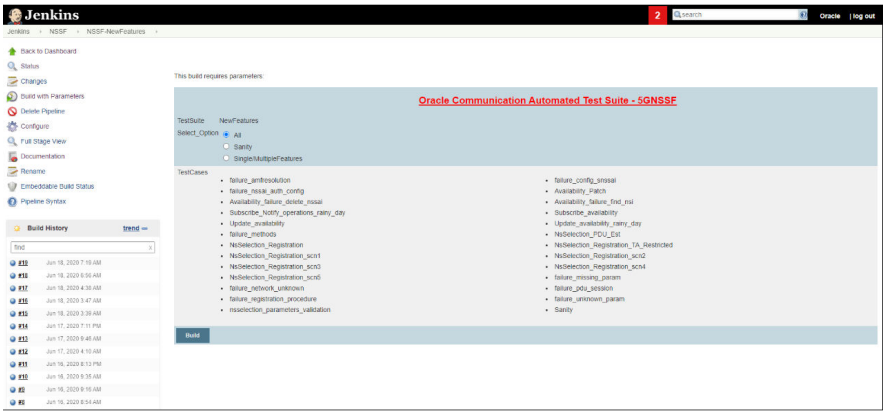
d. Click **Save** after making necessary changes. You are navigated back to the **Pipeline NSSF-NewFeatures** screen. Click **Build with Parameters** as shown below:

Figure 3-35 Build with Parameters



The following screen appears:

Figure 3-36 Build with Parameters Options

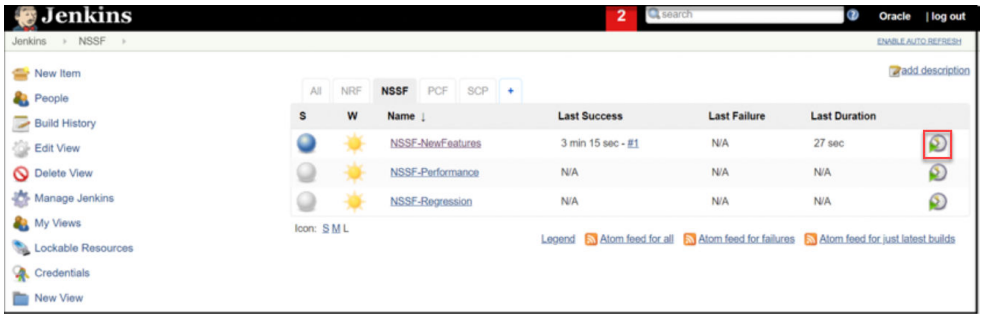


Executing NSSF Test Cases

To execute NSSF test cases:

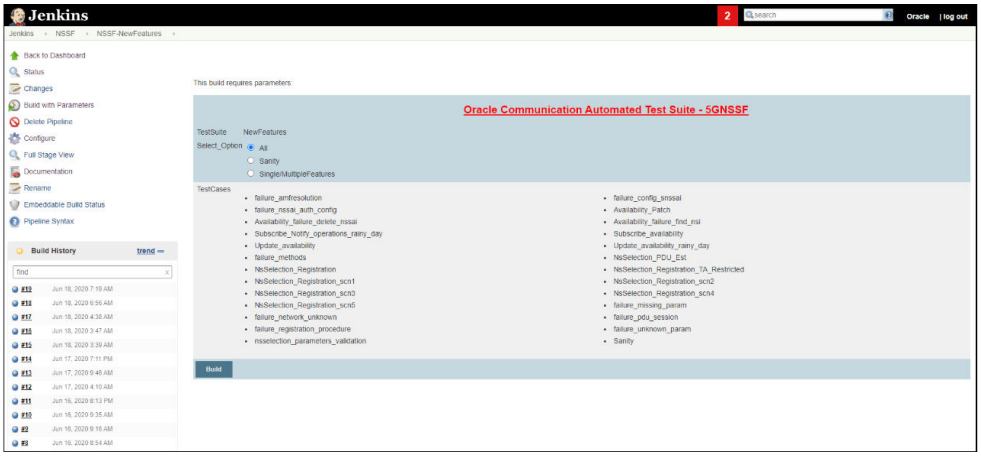
1. Click the **Schedule a Build with parameters** icon present on the NSSF-NewFeatures screen in the extreme right column corresponding to NSSF-NewFeatures row as shown below:

Figure 3-37 Schedule a Build with Parameters



2. The following screen appears:

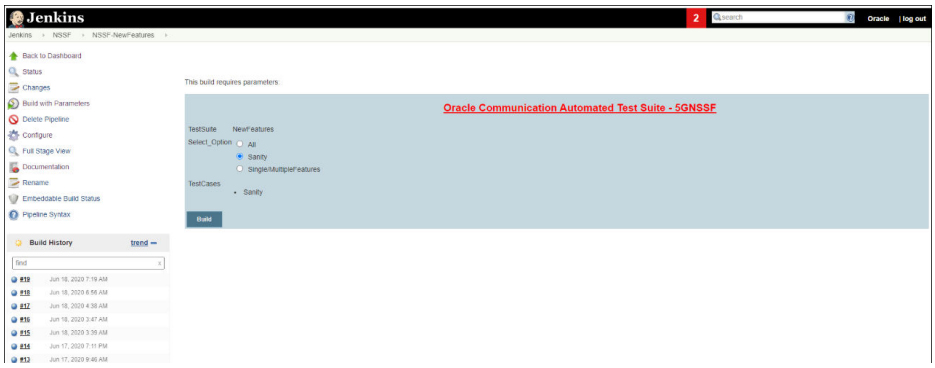
Figure 3-38 Build Screen



In the above screen, there are three **Select_Option(s)**, which are:

- **All:** By default, all the NSSF test cases are selected for execution. User just needs to scroll down and click **Build** to execute all the test cases.
- **Sanity:** It is recommended to execute Sanity before executing any test case. This helps to ensure that all the deployments are done properly or not. When you select Sanity, the following screen appears.

Figure 3-39 Select_Option(s) - Sanity



Click **Build** to execute all the sanity test cases.

- **Single/MultipleFeature:** This option allows you to select any number of test cases that you want to execute from the list of total test cases available for execution. After selecting the test cases, scroll-down and click **Build**. The selected NSSF test cases are executed.

The NSSF test cases are divided into NSSF Service operations as follows:

- **Availability Update:** These feature files are listed with a prefix as "Update".
- **Configuration:** These feature files are listed with a prefix as "failure".
- **Registration:** These feature files are listed with a prefix as "NsSelection_Registration".
- **PDU Session:** These feature files are listed with a prefix as "NsSelection_PDU".
- **NSSF Sanity:** This feature file contains all the basic sanity cases for NSSF ATS 1.6.1.
- **Subscription:** These feature files are listed with a prefix as "Subscribe".

NewFeatures - Documentation

To view NSSF functionalities, go to NSSF-NewFeatures pipeline and click the **Documentation** link in the left navigation pane. The following screen appears:

Figure 3-40 NSSF - Documentation

Each one of the documentation features is described below:

- **NSSF_BASIC_SANITY_CASES** - Lists all the sanity cases, which are useful to identify whether all the NSSF functionality works fine.

- **NSSF_CONFIG_CASES** - Lists all the test cases related to NSSF configuration.
- **NSSF_BASIC_UPDATE_CASES** - Lists all the test cases relaed to Availability Update.
- **NSSF_AVAILABILITY_PATCH_AND_NEGATIVE_CASES** - Lists all the test cases related to Availability Patch and other negative scenarios.
- **NSSF_NsSelection_REGISTRATION_CASES** - Lists all the test cases related to NsSelection registration.
- **NSSF_NsSelection_PDU_CASES** - Lists all the test cases related to NsSelection PDU related cases.
- **NSSF_BASIC_SUBSCRIBE_CASES** - Lists all the test cases related to subscription.

You can click any functionality to view its test cases and scenarios of each test case. A sample screen is given below:

Figure 3-41 NSSF_BASIC_SANITY_CASES

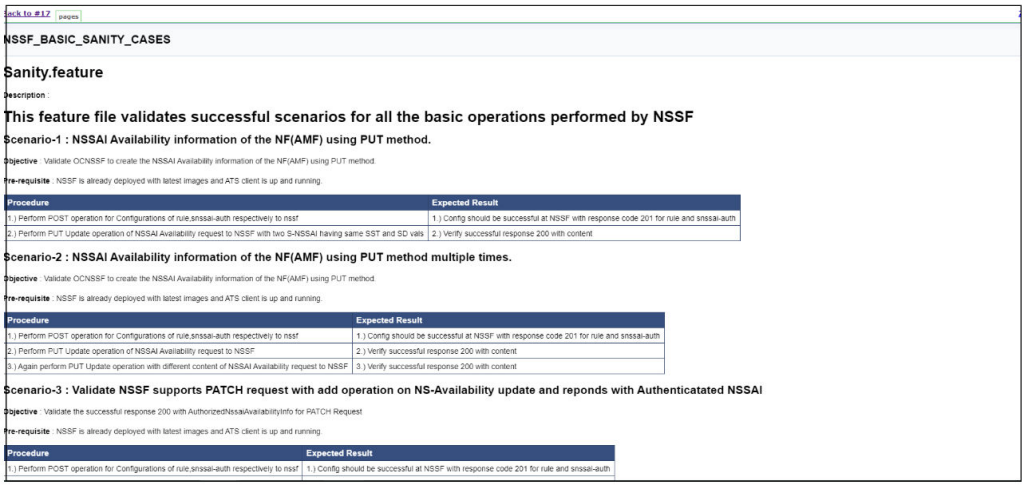


Figure 3-42 NSSF_BASIC_SUBSCRIBE_CASES

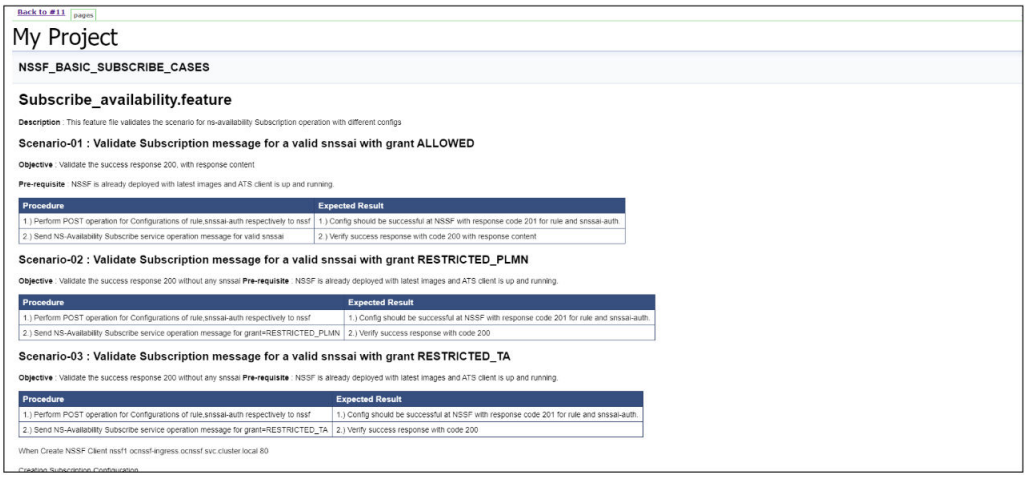
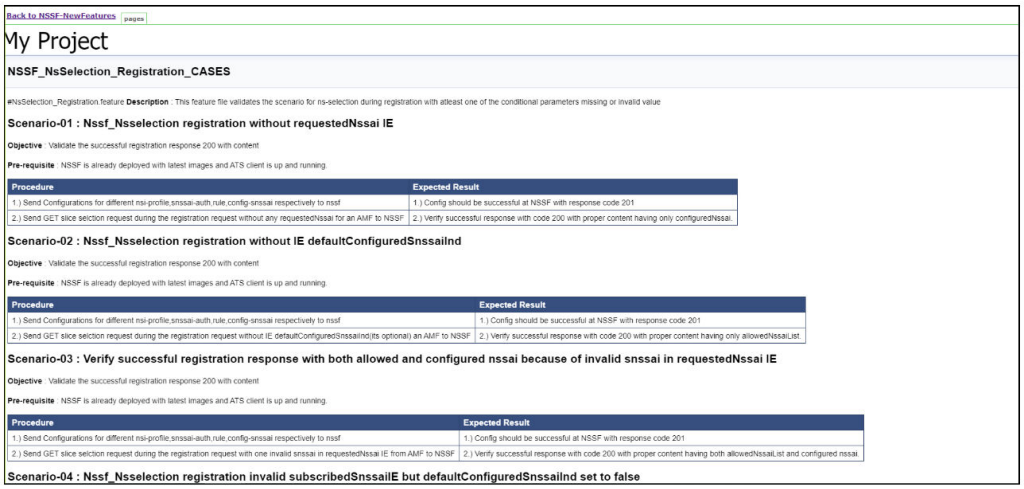


Figure 3-43 NSSF_NsSelection_Registration_CASES


Back to NSSF-NewFeatures pages

My Project

NSSF_NsSelection_Registration_CASES

#NsSelection_Registration.feature **Description** : This feature file validates the scenario for ns-selection during registration with atleast one of the conditional parameters missing or invalid value

Scenario-01 : Nssf_Nsselection registration without requestedNssai IE

Objective : Validate the successful registration response 200 with content

Pre-requisite : NSSF is already deployed with latest images and ATS client is up and running.

Procedure	Expected Result
1.) Send Configurations for different nsi-profile,snssai-auth,rule,config-snssai respectively to nssf	1.) Config should be successful at NSSF with response code 201
2.) Send GET slice selction request during the registration request without any requestedNssai for an AMF to NSSF	2.) Verify successful response with code 200 with proper content having only configuredNssai.

Scenario-02 : Nssf_Nsselection registration without IE defaultConfiguredSnssaiInd

Objective : Validate the successful registration response 200 with content

Pre-requisite : NSSF is already deployed with latest images and ATS client is up and running.

Procedure	Expected Result
1.) Send Configurations for different nsi-profile,snssai-auth,rule,config-snssai respectively to nssf	1.) Config should be successful at NSSF with response code 201
2.) Send GET slice selction request during the registration request without IE defaultConfiguredSnssaiInd(its optional) an AMF to NSSF	2.) Verify successful response with code 200 with proper content having only allowedNssaiList.

Scenario-03 : Verify successful registration response with both allowed and configured nssai because of invalid snssai in requestedNssai IE

Objective : Validate the successful registration response 200 with content

Pre-requisite : NSSF is already deployed with latest images and ATS client is up and running.

Procedure	Expected Result
1.) Send Configurations for different nsi-profile,snssai-auth,rule,config-snssai respectively to nssf	1.) Config should be successful at NSSF with response code 201
2.) Send GET slice selction request during the registration request with one invalid snssai in requestedNssai IE from AMF to NSSF	2.) Verify successful response with code 200 with proper content having both allowedNssaiList and configured nssai.

Scenario-04 : Nssf_Nsselection registration invalid subscribedSnssaiIE but defaultConfiguredSnssaiInd set to false

Executing Policy Test Cases using ATS

This ATS-Policy release is a converged release comprising of scenarios (test cases) from PCF, CN-PCRF and Converged Policy modes. ATS 1.3.1 is compatible with Policy 1.8.1 with TLS Enabled (server side) and Disabled Mode, CN-PCRF and Converged policy.

To execute Policy test cases, you need to ensure that following prerequisites are fulfilled.

Prerequisites

- Deploy OCCNP.
- Install Go-STUB and DNS-Bind stub for PCF and Converged Policy mode.
- ATS Prometheus Metrics validation works only when the installation has a single pod for each microservice in the CN Policy deployment.
- Users can customize test cases in the custom test case folders (cust_newfeatures, cust_regression and cust_performance). They can add new test cases, remove unwanted test cases and modify existing test cases. It does not impact the original product packaged test cases available in the newfeatures, regression and performance folders. For more details, you can refer to Custom Folder Implementation.
- In the **-application-config** configmap, configure the following parameters with the respective values:
  - `primaryNrfApiRoot=http://nf1stub.<namespace_gostubs_are_deployed_in>.svc:8080`
    Example: `primaryNrfApiRoot=http://nf1stub.ocats.svc:8080 #`
  - `secondaryNrfApiRoot=http://nf1stub.ocats.svc:8080 (comment out the secondaryNrfApiRoot)`
  - nrfClientSubscribeTypes=UDR,CHF
  - #supportedDataSetId=POLICY (comment out the supportedDataSetId)

> **Note:**
>
> You can configure these values at the time of Policy deployment.

> **Note:**
>
> To get all configmaps in your namespace execute:
> `kubectl get configmaps -n <Policy_namespace>`

- Configure DNS Stub for session retry feature validation. The steps are as follows:
  1. Login to DNS Stub.

     > **Note:**
     >
     > You can refer to the **Policy ATS Installation Procedure** chapter to learn the process to install DNS Stub.

     Example: `kubectl exec -it ocdns-ocdns-bind-86888f75cf-kv64w -n ocpcf bash`
  2. Edit the **named.conf.local** file present in /etc/bind location.
     a. Replace **occne15-ocpcf-ats** with **cluster name** where ocpcf is deployed.
     b. Replace **ocats** with the namespace where stubs are deployed.
     c. Replace **10.233.0.3** with the core DNS server IP address in your deployment. If the core DNS server IP address is not known, you can use the next available forwarder IP address. Execute the following command inside DNS stub pod to know the next available forwarder:

        ```
        cat /etc/resolv.conf
        ```

Figure 3-44 Editing named.conf.local

d. Go to **/etc/bind/zones** and edit each of these files; **db.udr**, **db.chf** and **db.smf**. Replace **ocats** with the namespace where stubs are deployed.

Figure 3-45 Editing db.udr, db.chf and db.smf files

```
$TTL    604800
@       IN      SOA     nsl.nflstub.ocats.svc. root.nflstub.ocats.svc. (
                      3         ; Serial
                 604800     ; Refresh
                  86400     ; Retry
                2419200     ; Expire
                 604800 )   ; Negative Cache TTL
;
; name servers - NS records
     IN      NS      nsl.nflstub.ocats.svc.

; name servers - A records
nsl.nflstub.ocats.svc.          IN      A       172.20.0.2

host1.nflstub.ocats.svc.        IN      A       172.20.0.3
host2.nflstub.ocats.svc.        IN      A       172.20.0.4

;# _service._proto.name.  TTL   class SRV priority weight port target.
_http._tcp.nflstub.ocats.svc.   86400 IN    SRV 1        60     8080 nflstub.ocats.svc.
_http._tcp.nflstub.ocats.svc.   86400 IN    SRV 20       20     8080 nfl1stub.ocats.svc.
_http._tcp.nflstub.ocats.svc.   86400 IN    SRV 10       20     8080 nfl2stub.ocats.svc.
```

3. After making all the changes, execute the following command to restart the bind service:
   `/etc/init.d/bind9 restart`

- Edit the **Alternate Route Service** deployment that points toward DNS Stub. By default, it points to CoreDNS with following settings in deployment file:

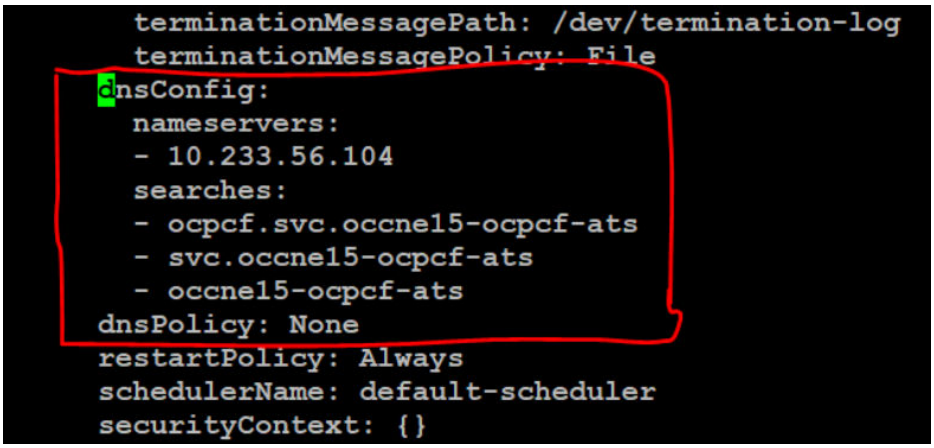
Figure 3-46 Alternate Route Service Deployment File

```
    terminationMessagePolicy: File
  dnsPolicy: ClusterFirst
  restartPolicy: Always
```

  - Execute the following command to add the given content in alternate service to query DNS stub:
    `kubectl edit deployment ocpcf-occnp-alternate-route -n ocpcf`
  - Add the IP Address of the nameservers that you have recorded after installing the DNS stub (cluster IP Address of DNS Stub).
  - Add the search function based on cluster name and namespace where you have deployed PCF:
    `occne15-ocpcf-ats → cluster name`

    `ocpcf → namespace where pcf deployed`
  - Set dnsPolicy to "None".

Figure 3-47 Policy Alternate Service - DNS Config Info



Once the DNS stub is configured, login to alternate service pod and execute the following commands to verify if the stubs are correctly configured:

```
[cloud-user@pcf-occ15-vcne-bastion-1 ~]$ kubectl exec -it ocpcf-
occnp-alternate-route-9cd8558c5-2vwnh -n ocpcf bash
```

```
[dnssrv@ocpcf-occnp-alternate-route-9cd8558c5-2vwnh
oracle]$ curl 'http://10.233.48.21:8000/lookup?
fqdn=nf1stub.ocpcf.svc&scheme=http' -X GET
```

```
[dnssrv@ocpcf-occnp-alternate-route-9cd8558c5-2vwnh
oracle]$ curl 'http://10.233.48.21:8000/lookup?
fqdn=nf2stub.ocpcf.svc&scheme=http' -X GET
```

```
[dnssrv@ocpcf-occnp-alternate-route-9cd8558c5-2vwnh
oracle]$ curl 'http://10.233.48.21:8000/lookup?
fqdn=nf3stub.ocpcf.svc&scheme=http' -X GET
```

where, 10.233.48.21 is an alternate service cluster IP.

Output of this command should be displayed as follows showing that the target information for the anchor FQDN is associated for the different stubs:

```
[{"target":"nf11stub.ocpcf.svc","port":8080,"ttl":86400,"type":"SRV"
,"dclass":"IN","priority":20,"weight":20}
,
{"target":"nf1stub.ocpcf.svc","port":8080,"ttl":86400,"type":"SRV","
dclass":"IN","priority":1,"weight":60}
,
{"target":"nf12stub.ocpcf.svc","port":8080,"ttl":86400,"type":"SRV",
"dclass":"IN","priority":10,"weight":20}]
```

- **PCF**

- **PCF with TLS not available:** In the PCF's custom values file, check if the following parameters are configured with the respective values:

```
ingress-gateway:
                enableIncomingHttps: false
        egress-gateway:
                enableOutgoingHttps: false
```

- **PCF with TLS Enabled:** In the PCF's custom values file, check if the following parameters are configured with the respective values:

```
 ingress-gateway:
                enableIncomingHttps: true
  egress-gateway:
                enableOutgoingHttps: true/false
```

  You also need to ensure that PCF is deployed with corresponding certificates.

  This scenario has two options:

  * **Client without TLS Enabled:** In this case, PCF is deployed with TLS enabled without generating any certificate in the ATS pod.
  * **Client with TLS Security Enabled:** In this case, PCF and ATS both have required certificates. For more details, refer to the **Enabling Https support for Egress and Ingress Gateway** section in this topic.

- In the **-application-config** configmap, configure the following parameters with the respective values:

  * primaryNrfApiRoot=http://nf1stub.<namespace_gostubs_are_deployed_in>.svc:8080
    Example: primaryNrfApiRoot=http://nf1stub.ocats.svc:8080
  * nrfClientSubscribeTypes=UDR,CHF
  * supportedDataSetId=POLICY (Comment out the supportedDataSetId)

  > **Note:**
  >
  > You can configure these values at the time of Policy deployment also.

  > **Note:**
  >
  > Execute the following command to get all configmaps in your namespace.
  > `kubectl get configmaps -n <Policy_namespace>`

- **CN-PCRF**
  Execute the following command to set the Log level to Debug in Diam-GW POD:

  `kubectl edit statefulset <diam-gw pod name> -n <namespace>`

  Execute the following command to set the default peer in the configuration map:

```
kubectl edit statefulset <diam-gw pod name> -n <namespace>
```

- name: USE_CFG_SVC

value: "false"

Execute the following command to edit and set the default configuration of Diameter peer in the diam configuration map and set the **responseOnly** parameter to true.

```
kubectl edit cm oc-diam-gateway-config-peers -n <namespace>
```

Figure 3-48 Setting Diameter Log Level Configuration

```
nodes:
  - name: 'ocpcf-occnp-pcrf-core'
    type: 'pcrf'
    responseOnly: true
    host: ocpcf-occnp-pcrf-core-headless
    port: 3868
    realm: ''
    identity: ''
```

- **Converged Policy:** It is same as PCF and CN-PCRF. You can refer to explanation given above.
- Prometheus server should be installed in cluster.
- Database cluster should be in a running state with all the required tables. You need to ensure that there are no previous entries in database before executing test cases.
- Deploy ATS in the same namespace as Policy using Helm Charts.
- User **MUST NOT** initiate a job in two different pipelines at the same time.
- The installation should have only one pod for each microservice related to ATS Prometheus Metrics validation to work in the CN Policy deployment.
- If you enable Service Mesh check, then you need to create a destination rule for fetching the metrics from the Prometheus. In most of the deployments, Prometheus is kept outside the service mesh so you need a destination rule to communicate between TLS enabled entity (ATS) and non-TLS entity (Prometheus). You can create a destination rule as follows:

```
kubectl apply -f - <<EOF

apiVersion:networking.istio.io/v1alpha3
kind:DestinationRule
metadata:
  name:prometheus-dr
  namespace:ocats
spec:
  host:oso-prometheus-server.pcf.svc.cluster.local
  trafficPolicy:
    tls:
```

```
      mode:DISABLE
EOF
```

In the destination rule:

- **name** indicates the name of destination rule.
- **namespace** indicates where the ATS is deployed.
- **host** indicates the hostname of the prometheus server.

Enabling TLS in ATS Pod

You can enable TLS in ATS pod after successful deployment of PCF (TLS enabled server side) and ATS. To enable TLS in ATS Pod:

1. Execute the following command to copy the caroot.cer generated while PCF deployment to ATS pod in "cert" directory.

   ```
   kubectl cp  <path_to_file>/caroot.cer <namespace>/<ATS-Pod-name>:
   /var/lib/jenkins/cert/ -n <namespace>
   ```

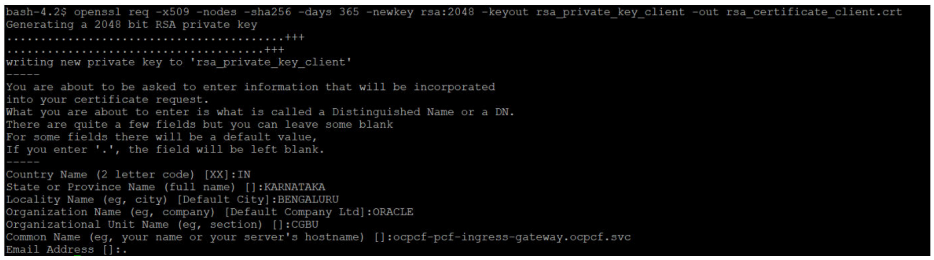
   Example:

   ```
   kubcetl cp cert/caroot.cer ocpcf/ocpcf-ocats-pcf-56754b9568-rkj8z:
   /var/lib/jenkins/cert/
   ```

2. Execute the following command to login to your ATS Pod.

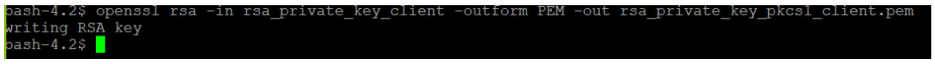
   ```
   kubectl exec -it <ATS-Pod-name> bash -n <namespace>
   ```

3. Execute the following commands from cert directory to create private key and certificates:

   a. ```
      openssl req -x509 -nodes -sha256 -days 365 -newkey rsa:2048
      -keyout
      rsa_private_key_client -out rsa_certificate_client.crt
      ```

   **Figure 3-49 Command 1**




> **Note:**
>
> You need to provide appropriate values and specify fqdn of PCF Ingress Gateway service i.e. <ingress-servicename>.<pcf_namespace>.svc in Common Name.
      ```

b. `openssl rsa -in rsa_private_key_client -outform PEM -out rsa_private_key_pkcs1_client.pem`

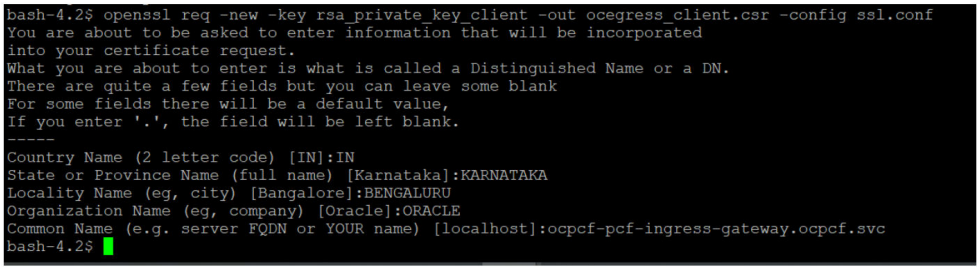
Figure 3-50 Command 2



c. `openssl req -new -key rsa_private_key_client -out ocegress_client.csr -config ssl.conf`

> **Note:**
>
> You can either use or copy the ssl.conf file, which was used while deploying PCF to ATS pod for this step.

Figure 3-51 Command 3



4. Execute the following command to copy the ocegress_client.csr to the bastion.

```
openssl x509 -CA caroot.cer -CAkey cakey.pem -CAserial serial.txt
-req
      -in ocegress_client.csr -out ocegress_client.cer -days 365
-extfile
    ssl.conf -extensions req_ext
```

Figure 3-52 Copying ocegress_client.csr to bastion

5. Copy the **ocegress_client.cer** from Bastion to the ATS Pod.
6. Restart the ingress and egress gateway pods from the Bastion.

Logging into ATS

Before logging into ATS Jenkins GUI, it is important to get the nodeport of the service, 'ocats-Policy'. Execute the following command to get the nodeport:

```
kubectl get svc -n <Policy_namespace>
```

Example: `kubectl get svc -n ocpcf`

Figure 3-53 Policy Nodeport

```
ocats-ocats-pcf                LoadBalancer   10.233.56.56    10.75.225.49    8080:32471/TCP
```

To login to Jenkins, open the Web Browser and type the URL: http://<Worker-Node-IP>:<Node-Port-of-ATS>. In the above screen, 32471 is the nodeport. **Example:** http://10.75.225.49:32471

> **Note:**
>
> For more information on ATS deployment in PCF, refer to Policy ATS Installation Procedure.

Executing ATS

To execute ATS:

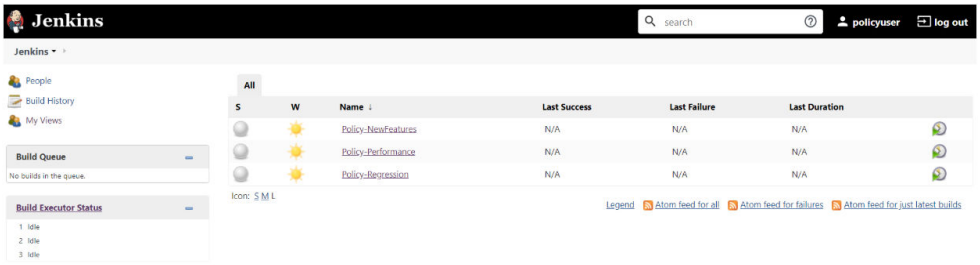
1. Enter the **username** as "policyuser" and **password** as "policypasswd". Click **Sign in**.

   > **Note:**
   >
   > If you want to modify your default login password, refer to Modifying Login Password

   The following screen appears showing policy pre-configured pipelines:

   - **Policy-NewFeatures:** This pipeline has all the test cases, which are delivered as part of Policy ATS.
   - **Policy-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
   - **Policy-Regression:** This pipleine has all the test cases, which were delivered in Policy ATS - 1.7.4

Figure 3-54 Pre-Configured Pipelines



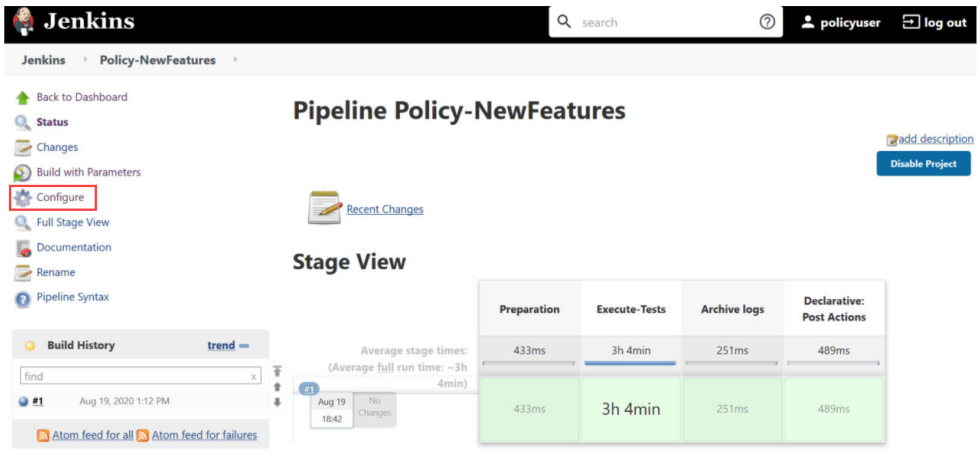
The pre-configured pipelines are explained below:

Policy-New Features Pipeline

This is a pre-configured pipeline where all the Policy new test cases are executed. To configure its parameters, which is a one time activity:

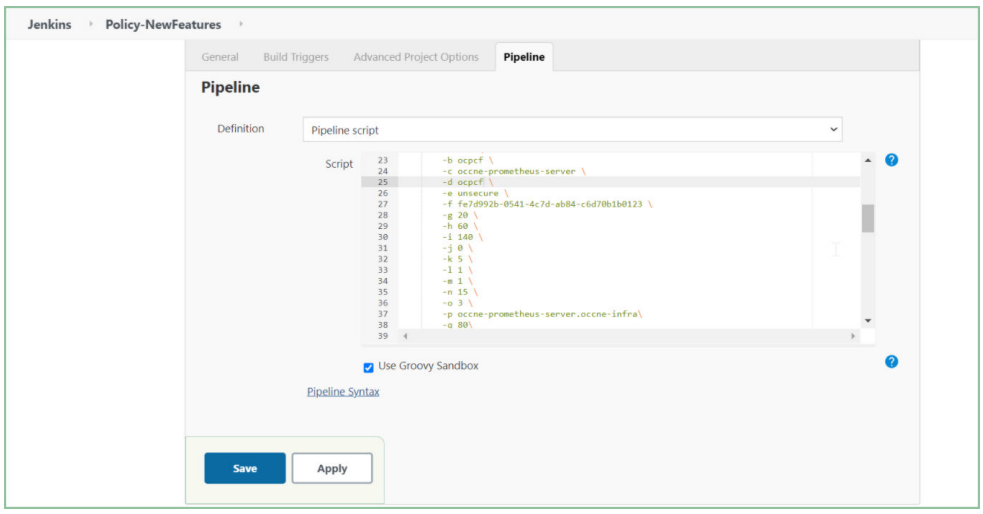
1. Click **Policy-NewFeatures** in the Name column and then, click **Configure** in the left navigation pane as shown below:

Figure 3-55 Policy-NewFeatures Configure



2. The Policy-NewFeatures, **General** tab appears. Make sure that the screen loads completely.
3. Scroll-down to the end. The control moves from **General** tab to the **Pipeline** tab as shown below:

Figure 3-56 Policy - Pipeline Script



In the **Script** area of the Pipeline section, you can change value of the following parameters:

- **b:** Change this parameter to update the namespace where Policy was deployed in your bastion.

- **d:** Change this parameter to update the namespace where your gostubs are deployed in your bastion.
- **e:** Set this parameter as 'unsecure', if you intend to run ATS in TLS disabled mode. Else, set this parameter as 'secure'.
- **g:** Set this parameter to more than 35 secs. The default wait time for the pod is 35 secs. Every TC requires restart of the nrf-client-management pod.
- **h:** Set this parameter to more than 60 secs. The default wait time to add a configured policy to the database is 60 secs.
- **i:** Set this parameter to more than 140 secs. The default wait time for Nf_Notification Test Cases is given as 140 secs.
- **k:** Use this parameter to set the waiting time to initialize Test Suite.
- **l:** Use this parameter to set the waiting time to get response from Stub.
- **m:** Use this parameter to set the waiting time after adding Policy Configuration.
- **n:** Use this parameter to set the waiting time after adding Policy.
- **o:** Use this parameter to set the waiting time before sending next message.
- **p:** Use this parameter to set Prometheus Server IP.
- **q:** Use this parameter to set Prometheus Server Port.

> **Note:**
>
> **DO NOT MODIFY ANYTHING OTHER THAN THESE PARAMETER VALUES.**

- Click **Save** after updating the parameters value. The Policy-NewFeatures Pipeline screen appears.

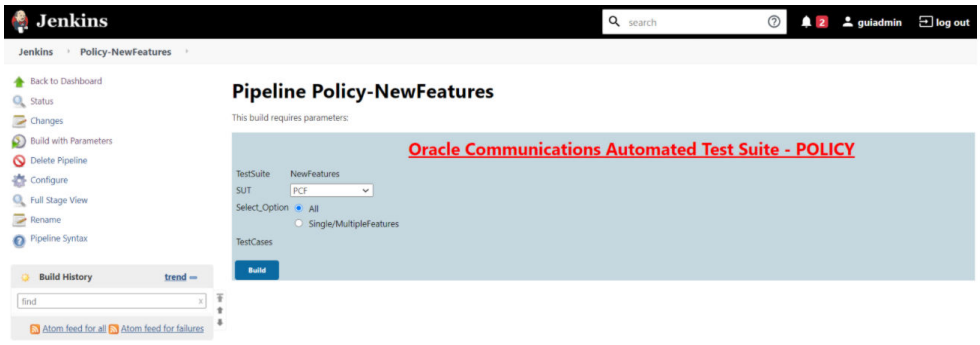
> **Note:**
>
> It is advisable to save the pipeline script in your local machine that you can refer at the time of ATS pod restart.

Executing Policy Test Cases

To execute Policy test cases:

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

Figure 3-57 Policy - Build with Parameters

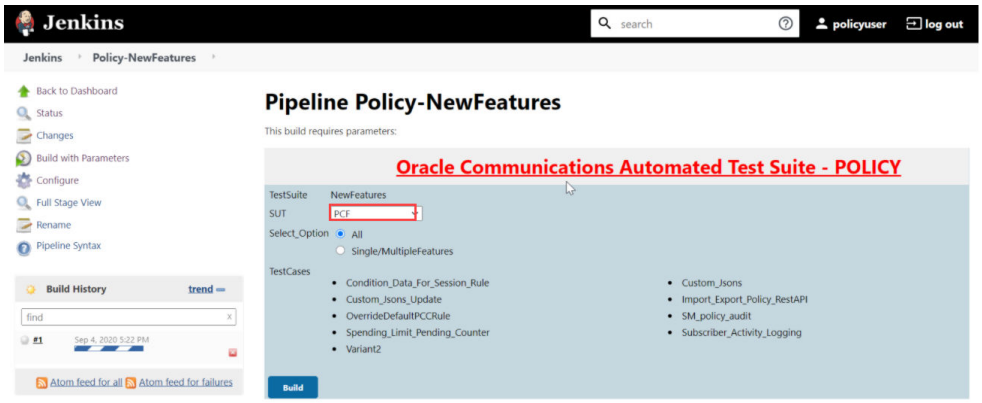


Note:

Jenkins display empty TestCases list as it is referring to the **custom** folder for Policy NewFeatures.

Copy the required test cases that are available in the PCF/PCRF/Common folder and place them in the appropriately within **custom** folder for Policy-NewFeatures. Reload the Jenkins page to view the cases available in the **custom** NewFeatures folder.

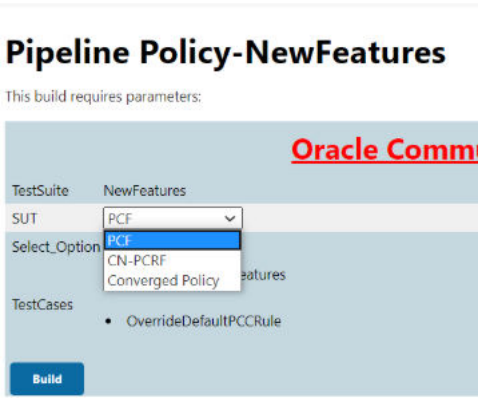
Figure 3-58 Policy - Viewing Custom test Cases



In the above screen, you can select **SUT** as either PCF, CN-PCRF or Converged Policy. It also has two **Select_Option**(s), which are:

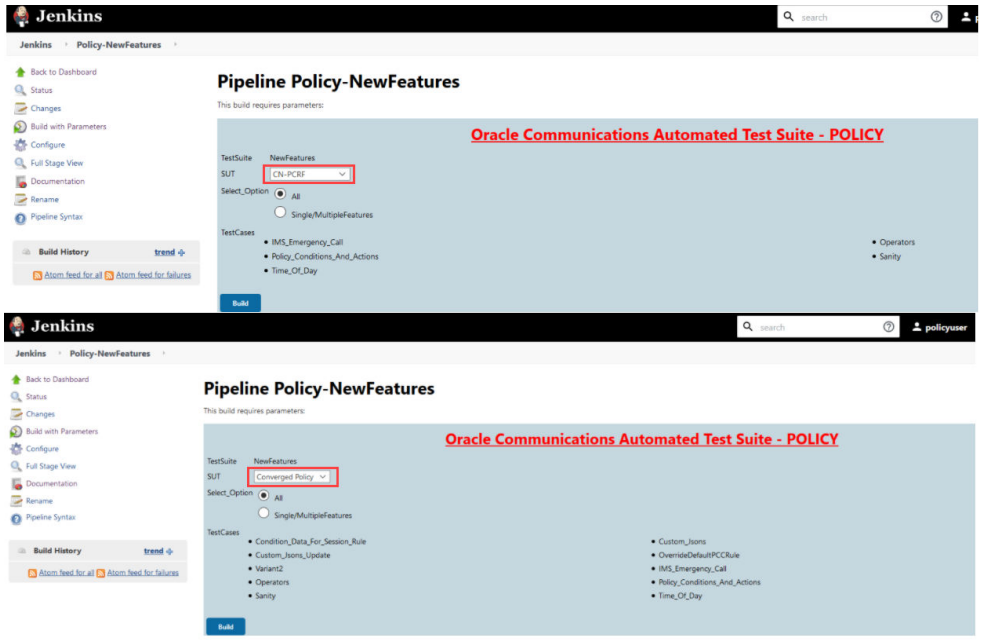
- **All:** By default, all the Policy test cases are selected for execution. Scroll down and click **Build** to execute all the test cases.
- **Single/MultipleFeatures:** This option allows you to select any number of test cases that you want to execute from the list of total test cases available for execution. After selecting the test cases, scroll-down and click **Build**. The selected Policy test cases are executed.

Figure 3-59 SUT Options



Based on your selection, related **TestCases** appear.

Figure 3-60 Test Cases based on SUT

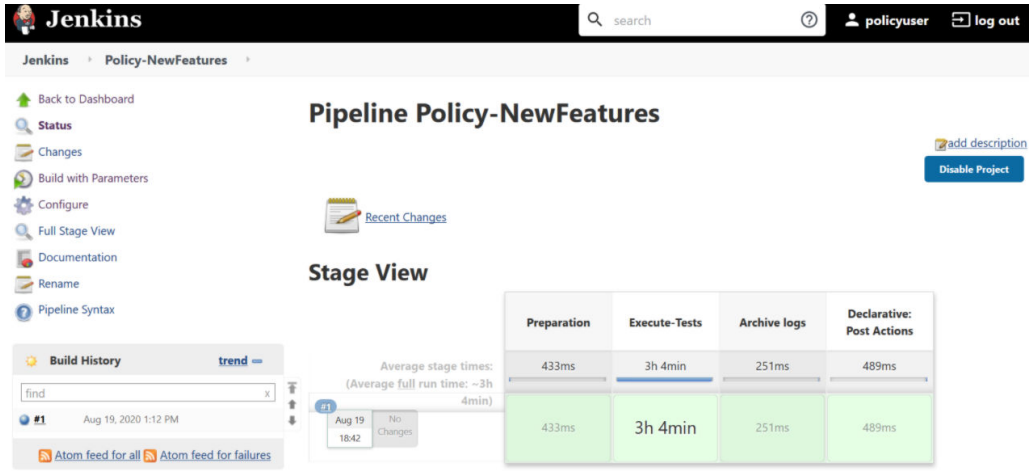


> **Note:**
>
> Converged Policy Test cases are combination of PCF and CN-PCRF TestCases.

Go to **Build** → **Console Output** to view the test result output as shown below:

Figure 3-61 Sample: Test Result Output in Console

Figure 3-62 Sample Output of Build Status



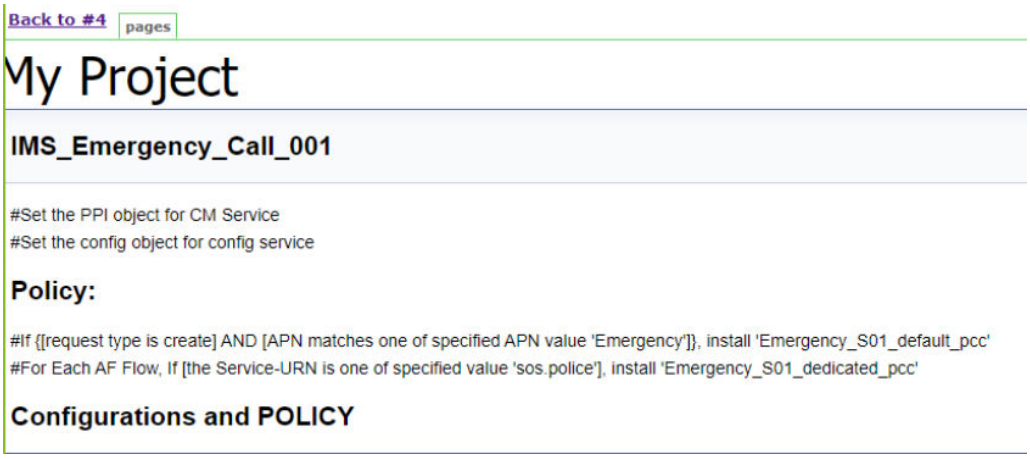
NewFeatures - Documentation

To view Policy functionalities, go to Policy-NewFeatures pipeline and click **Documentation** link in the left navigation pane.

Figure 3-63 Policy-NewFeatures Feature List

You can click any functionality to view its test cases and scenarios of each test case. For example, on click of IMS_Emergency_Call_001, the following test description appears:

Figure 3-64 IMS_Emergency_Call_001



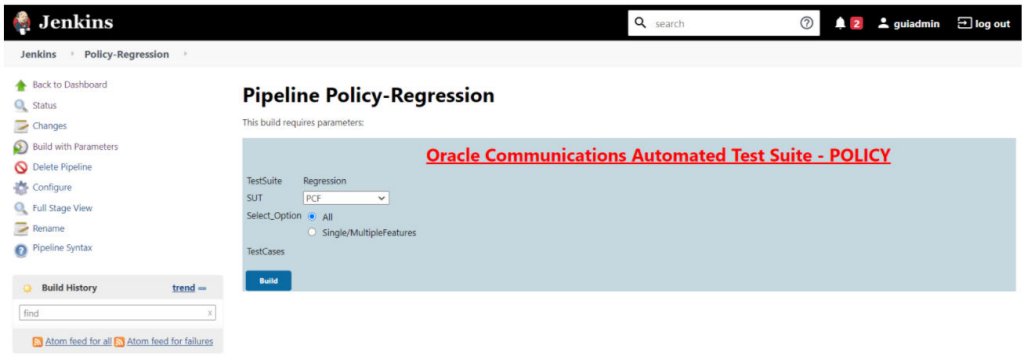
Based on the functionalities covered under Documentation, the **Build Requires Parameters** screen displays test cases. To navigate back to the Pipeline Policy-NewFeatures screen, click **Back to Policy-NewFeatures** link available on top left corner of the screen.

PCF-Regression Pipeline

This pre-configured pipeline has all the test cases of previous releases. For example, as part of Release 1.8.0, this pipeline has all the test cases that were released as part of release 1.7.4

To view Regression pipeline details, click **Build with Parameters** in the left navigation pane. The following screen appears:

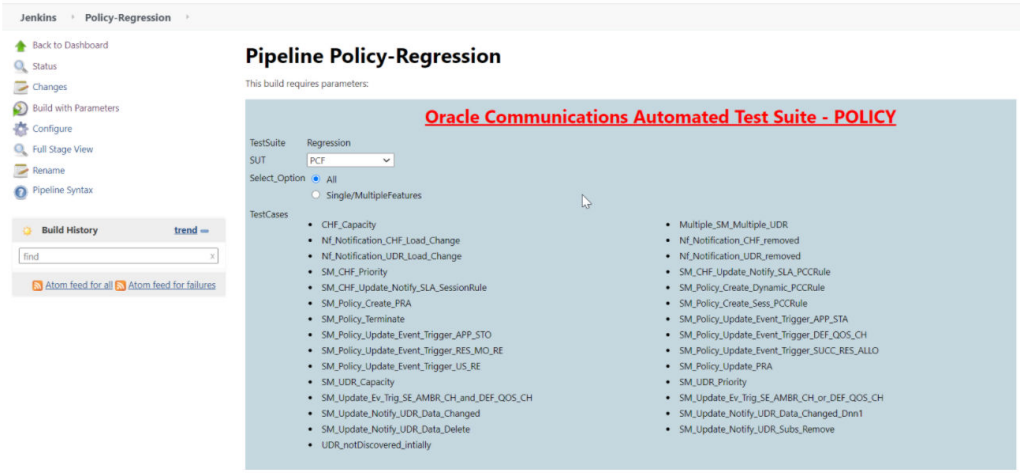
Figure 3-65 Policy-Regression



> **Note:**
>
> Jenkins display empty TestCases list as it is referring to the **custom** folder for Policy-Regression pipeline.

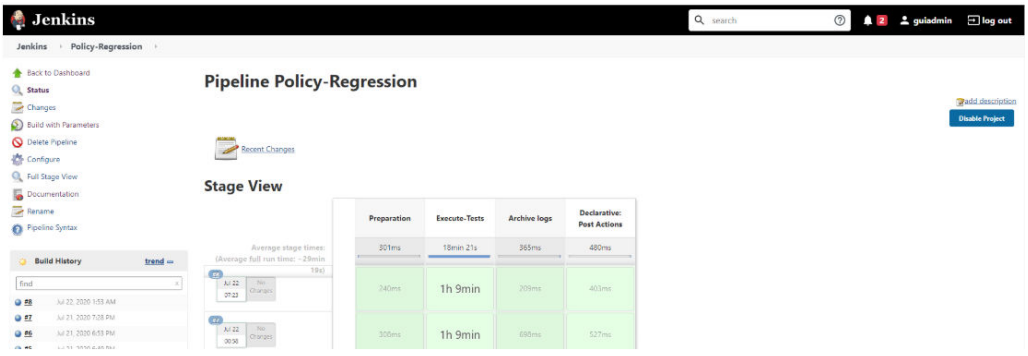
Copy the required test cases that are available in the PCF/PCRF/Common folder and place them in the appropriately within **custom** folder for Policy-Regression. Reload the Jenkins page to view the cases available in the **custom** Regression folder.

Figure 3-66 Policy - Regression - Viewing Custom Test Cases



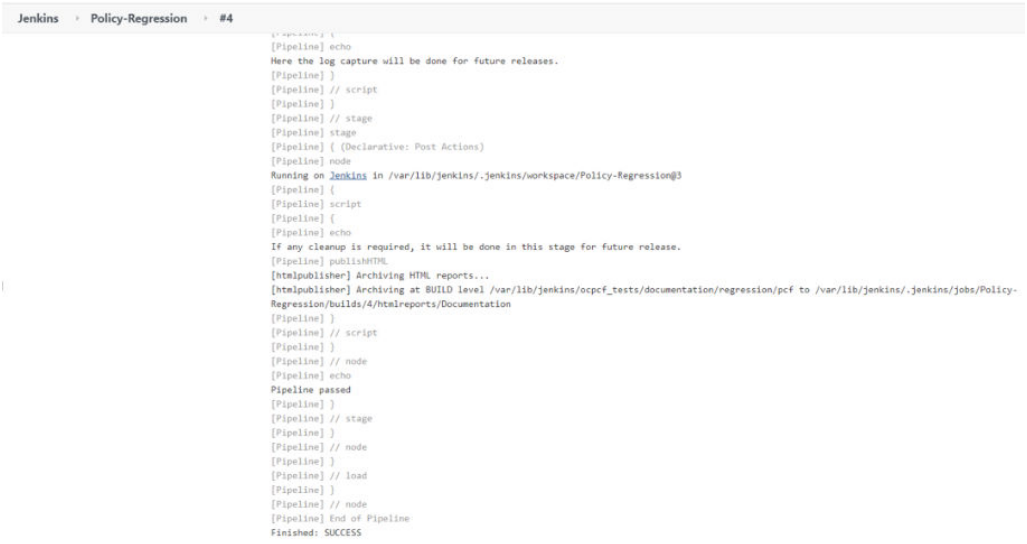
Click **Build**. The build output appears as shown below:

Figure 3-67 Policy-Regression Build Output



The console output is as follows:

Figure 3-68 Policy-Regression Console Output



> **Note:**
>
> The regression pipeline does not have any sanity option. However, you should perform all the steps as performed in NewFeatures pipeline. Configure the pipeline script changes to provide environment variables.

Regression - Documentation

Click **Documentation** in the left navigation pane of the Policy-Regression pipeline. Following screen appears:

Figure 3-69 Policy-Regression Documentation

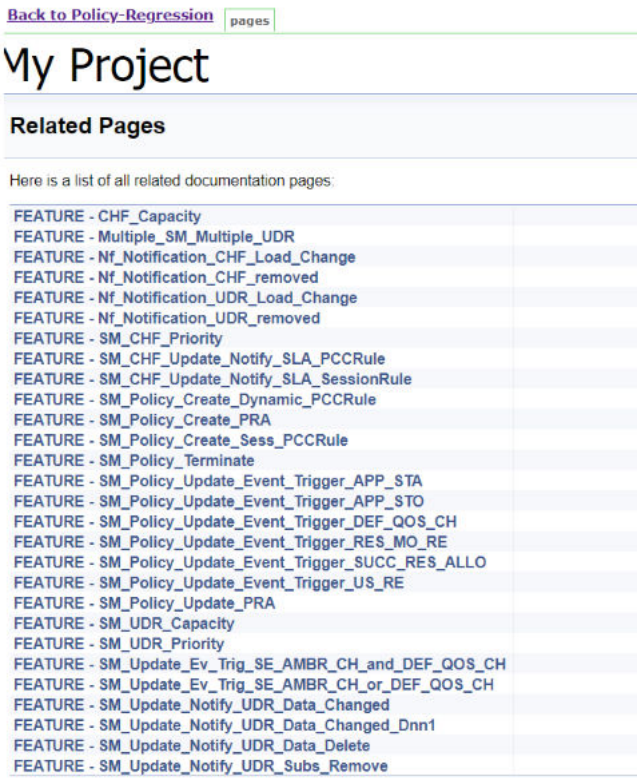
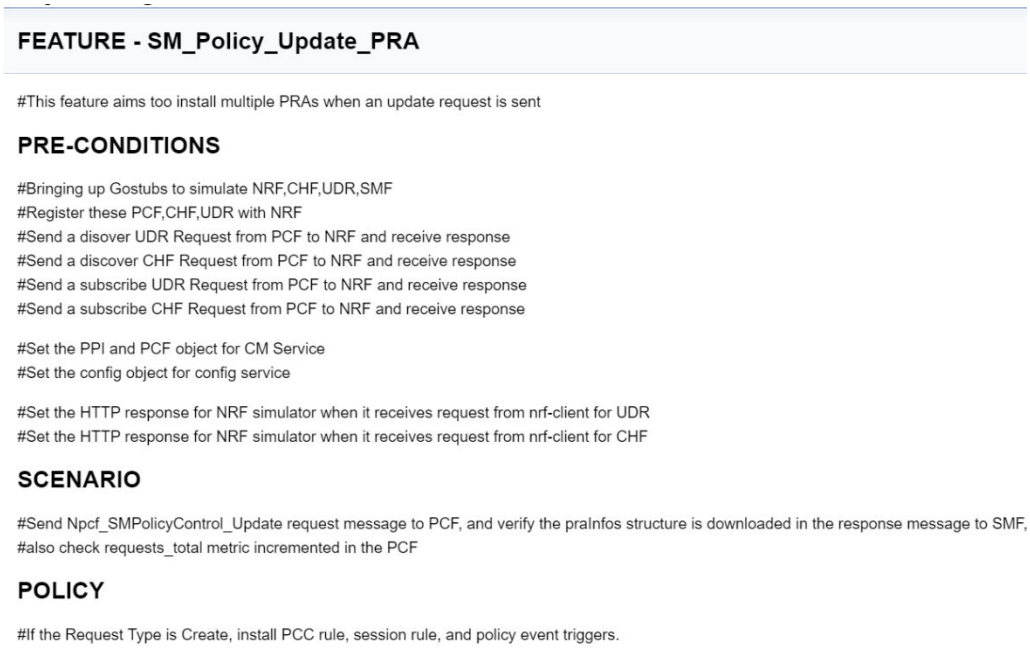


Figure 3-70 Sample: Regression Documentation - Feature



This screen shows functionalities of only those test cases that are released in previous releases.

Executing SCP Test Cases using ATS

To execute SCP Test Cases, you need to ensure that following prerequisites are fulfilled.

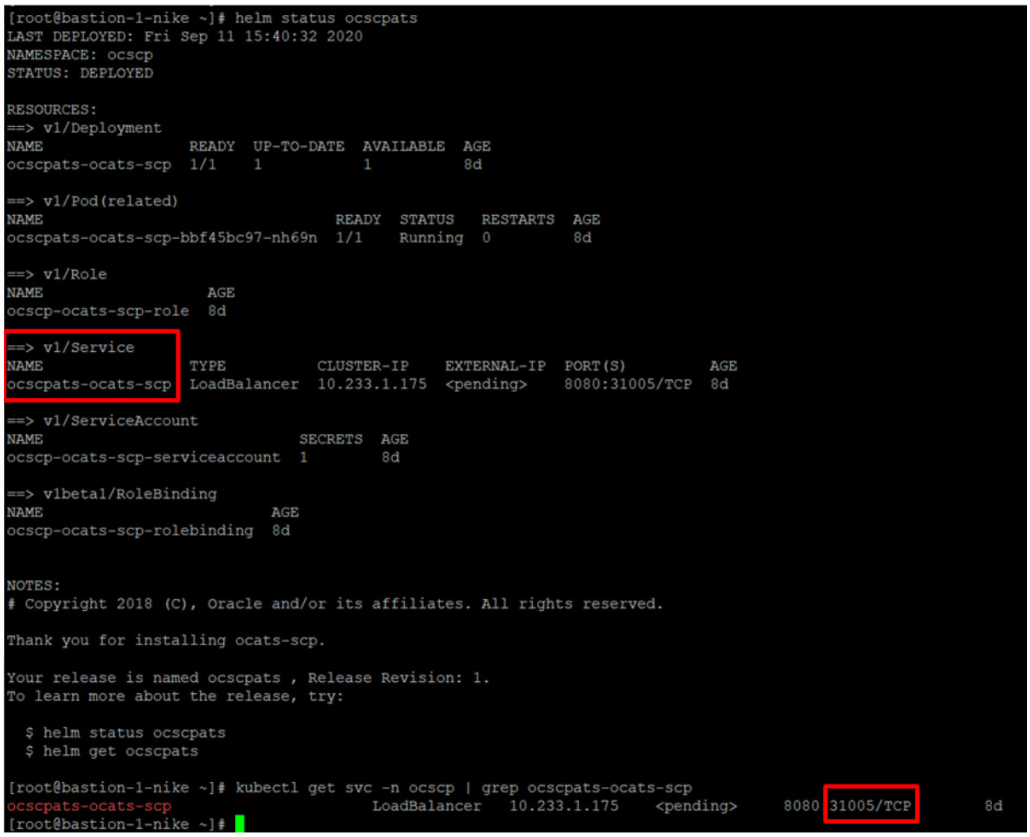
Prerequisites

- Deploy SCP 1.8.0 with following custom values in deployment file.
  - As you can provide NRF information only at the time of deployment, Stub NRF details like nrf1svc and nrf2svc should also be provided at the time of deployment before executing these cases. **For Example:** If teststub namespace is scpsvc then SCP should have been deployed with primary nrf as *nrf1svc.scpsvc.svc.<clusterDomain>* and secondary nrf as *nrf2svc.scpsvc.svc.<clusterDomain>* for NRF test cases to work.
  - Deploy NRF stubs with port 8080. Thus, NRF details of SCP should specify **ipEndPoints** port as 8080 without any **ipv4Address** field. **Example:** ipEndPoints: [{"port": "8080"}]).
  - In the SCP deployment file, **servingScope** should be 'Reg1', **servingLocalities** should have 'USEast' and 'Loc9'. In addition, the recommended **auditInterval** is '120' and **guardTime** is '10'.
  - For ATS execution, you should deploy SCP with SCP-Worker replicas set to 1.
- Users can customize test cases in the custom test case folders (cust_newfeatures, cust_regression and cust_performance). They can add new test cases, remove unwanted test cases and modify existing test cases. It does not impact the original product packaged test cases available in the newfeatures, regression and performance folders. For more details, you can refer to Custom Folder Implementation.
- Deploy ATS using helm charts.
- As you can deploy default ATS with role binding, it is important to deploy ATS and test stubs in the same namespace as SCP.

Logging into ATS

Before logging into ATS, you need to ensure that ATS is deployed successfully using HELM charts. A sample screen is given below:

Figure 3-71 Verifying ATS Deployment



To login to ATS Jenkins GUI, open the browser and provide the external IP of the worker node and nodeport of the ATS service as `<Worker-Node-IP>:<Node-Port-of-ATS>`. The Jenkins login screen appears.

> **Note:**
>
> In the **Verifying ATS Deployment** screen, the ATS nodeport is highlighed in red as **31005**. For more details on ATS deployment, refer toSCP ATS Installation Procedure .

Executing ATS

To execute ATS:

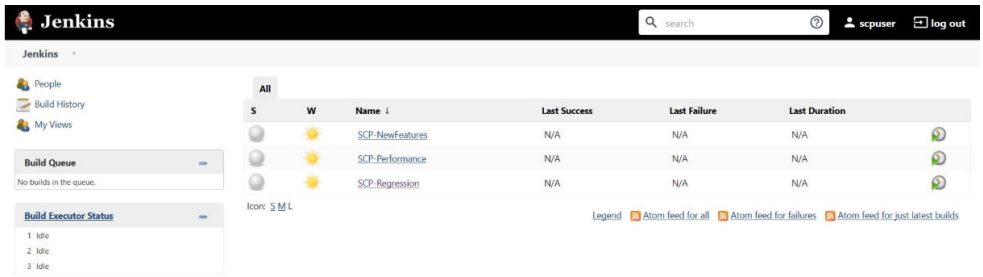
1. Enter the **username** as "scpuser" and **password** as "scppasswd". Click **Sign in**. A sample screen is shown below.

Figure 3-72 Logging into ATS GUI

> **Note:**
>
> If you want to modify your default login password, refer to Modifying Login Password

2. Following screen appears showing pre-configured pipelines for SCP individually (3 Pipelines).
   - **SCP-New-Features:** This pipeline has all the test cases, which are delivered as part of SCP ATS - 1.8.0
   - **SCP-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
   - **SCP-Regression:** This pipeline covers all the test cases of the previous releases.

Figure 3-73 ATS SCP First Logged-In Screen



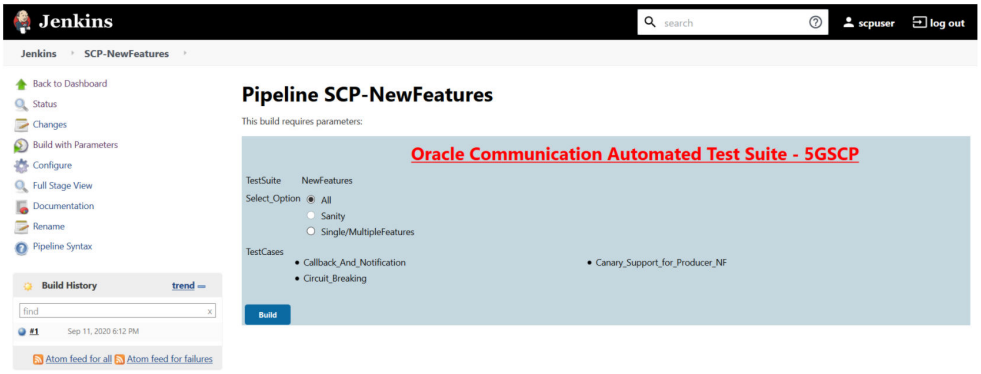
Pipeline SCP-NewFeatures

This is a pre-configured pipeline where all the SCP test cases are executed. If you are executing SCP pipeline for the first time then you have to set the Input Parameters before executing any test case. There is no need to set these parameters again unless there is any change in the configuration.

To configure its parameters, which is a one time activity:

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

Figure 3-74 SCP-NewFeatures

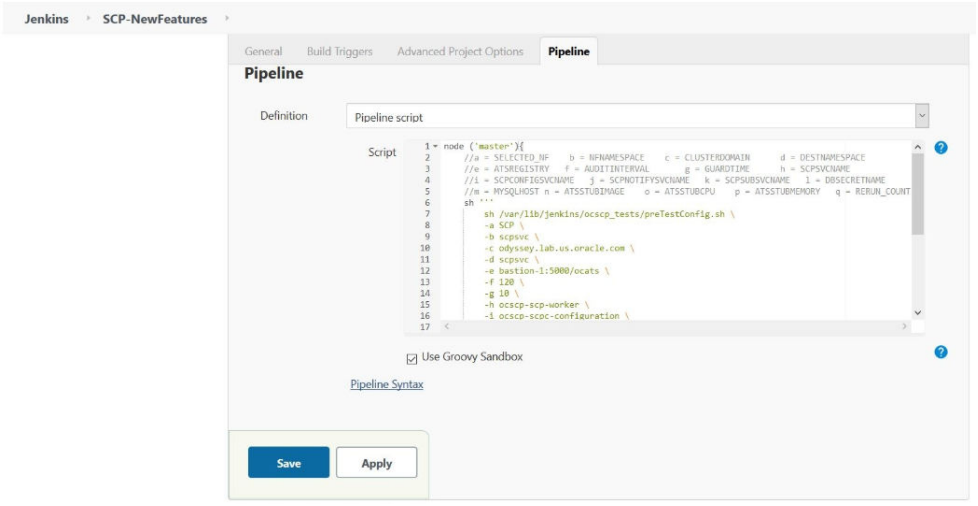


2. Click **Configure** in the left navigation pane to provide input parameters. The SCP-NewFeatures Configure - General tab appears.

> **Note:**
>
> **MAKE SURE THAT THE SCREEN SHOWN BELOW LOADS COMPLETELY BEFORE YOU PERFORM ANY ACTION ON IT. ALSO, DO NOT MODIFY ANY CONFIGURATION OTHER THAN DISCUSSED BELOW.**

3. Scroll-down to the end. The control moves from **General** tab to the **Pipeline** tab as shown below:

Figure 3-75 Pipeline Tab



You can modify script pipeline parameters from **"-b"** to **"-q"** on the basis of your deployment environment and click **Save**. The content of the pipeline script is as follows:

Figure 3-76 SCP Pipeline Content

```
node ('master'){
    //a = SELECTED_NF    b = NFNAMESPACE    c = CLUSTERDOMAIN      d = DESTNAMESPACE
    //e = ATSREGISTRY   f = AUDITINTERVAL       g = GUARDTIME      h = SCPSVCNAME
    //i = SCPCONFIGSVCNAME   j = SCPNOTIFYSVCNAME   k = SCPSUBSVCNAME   l = DBSECRETNAME
    //m = MYSQLHOST n = ATSSTUBIMAGE    o = ATSSTUBCPU    p = ATSSTUBMEMORY   q = RERUN_COUNT
    sh '''
        sh /var/lib/jenkins/ocscp_tests/preTestConfig.sh \
        -a SCP \
        -b scpsvc \
        -c odyssey.lab.us.oracle.com \
        -d scpsvc \
        -e bastion-1:5000/ocats \
        -f 120 \
        -g 10 \
        -h ocscp-scp-worker \
        -i ocscp-scpc-configuration \
        -j ocscp-scpc-notification \
        -k ocscp-scpc-subscription \
        -l cred \
        -m mysql.default \
        -n ocats-gostub:1.7.0 \
        -o 0.2 \
        -p 0.1G \
        -q 0
    '''
    load "/var/lib/jenkins/ocscp_tests/jenkinsData/Jenkinsfile-NewFeatures"
```

The description of these parameters is as follows:

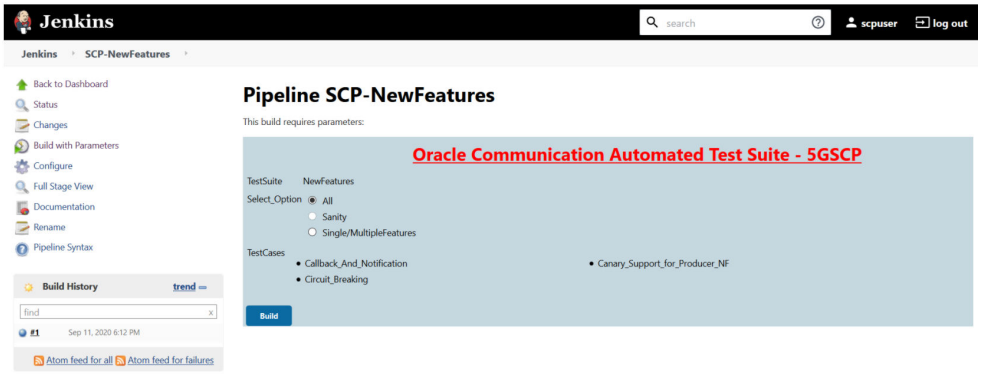
- -a - Selected NF
- -b - NameSpace in which SCP is Deployed
- -c - Kubernetes Cluster Domain where SCP is Deployed
- -d - Test Stubs NameSpace - must be same as SCP Namespace
- -e - Docker registry where test stub image is available
- -f - Audit Interval provided in SCP Deployment file
- -g - Guard Time provided SCP Deployment file
- -h - SCP-Worker microservice name as provided during deployment
- -i - SCPC-Configuration microservice name as provided during deployment
- -j - SCPC-Notification microservice name as provided during deployment
- -k - SCPC-Subscription microservice name as provided during deployment
- -l - DB Secret name as provided during deployment

- -m - Mysql Host name as provided during deployment
- -n - Test Stub Image Name with tag
- -o - Test Stub CPU requests and limit
- -p - Test Stub Memory requests and limit
- -q - re-run count

> **Note:**
>
> DO NOT MODIFY ANYTHING OTHER THAN THESE PARAMETERS.

4. Click the **Build with Parameters**. Following screen appears:

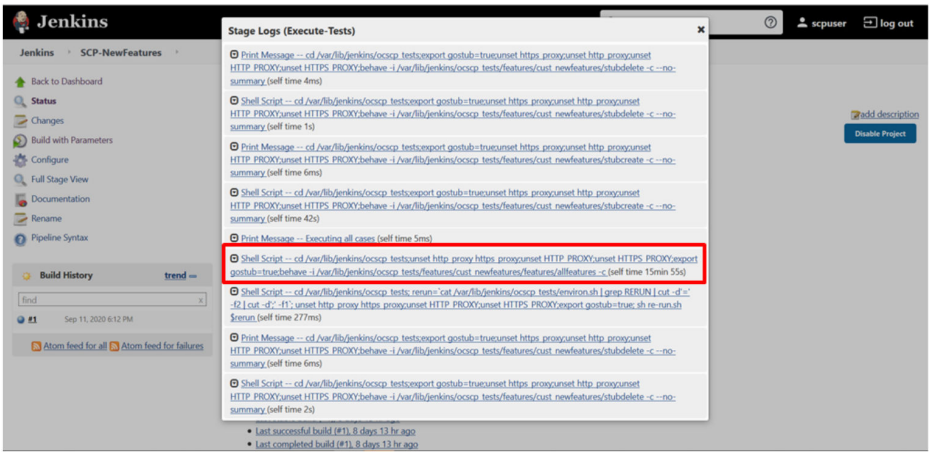
Figure 3-77 Build with Parameters Options



In the above screen, there are three **Select_Option**(s), which are:

- **All:** By default, all the SCP test cases are selected for execution. User just need to scroll down and click **Build** to execute all the test cases.
- **Sanity:** This option is **NOT AVAILABLE** for SCP.
- **Single/MultipleFeatures:** This option allows you to select any number of test cases that you want to execute from the list of total test cases available for execution. After selecting the test cases, scroll-down and click **Build**. The selected SCP test cases are executed.
- To check execution results and logs:
  - Click the execute-tests stage of pipeline and then logs.
  - Select the test execution step.
  - Double-click to open the execution logs console.

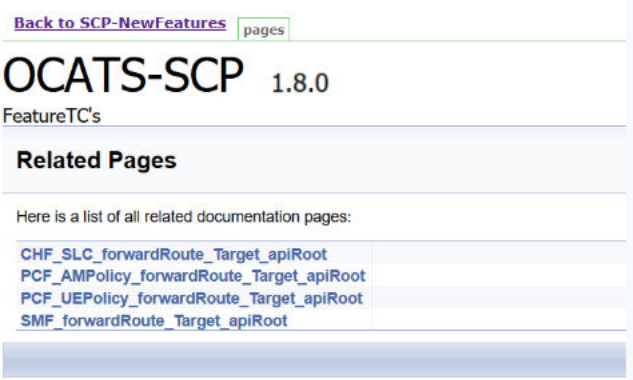
Figure 3-78 SCP-NewFeatures Stage Logs



NewFeatures - Documentation

This pipeline has the HTML report of all the feature files that you can test as part of SCP ATS release. To view SCP functionalities, go to SCP-NewFeatures pipeline and click **Documentation** link in the left navigation pane. The following screen appears:

Figure 3-79 SCP-NewFeatures-Documentation



> **Note:**
>
> Documentation option appears only if New-Features pipeline is executed atleast once.

You can click any functionality to view its test cases and scenarios for each test case. For example, on click of SMF_forwardRoute_Target_apiRoot, the following screen appears:

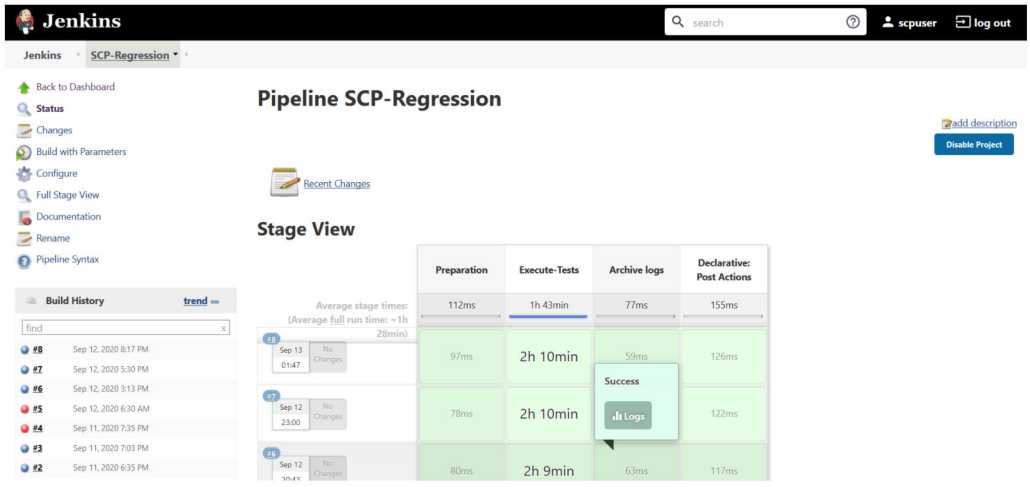
Figure 3-80 Sample: SCP Functionality

Based on the functionalities covered under Documentation, the **Build Requires Parameters** screen displays test cases. To navigate back to the Pipeline SCP-NewFeatures screen, click **Back to SCP-NewFeatures** link available on top left corner of the screen.

SCP-Regression Pipeline

This pre-configured pipeline has all the test cases of previous releases. When you click SCP-Regression Pipeline, following screen appears:

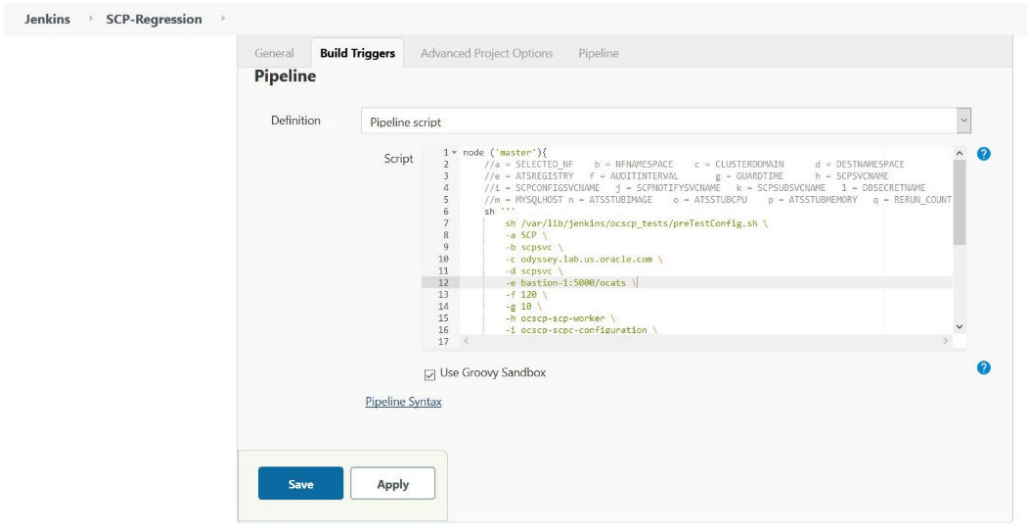
Figure 3-81 SCP-Regression Pipeline



If you are executing SCP pipeline for the first time, you have to set the Input Parameters before execution. Subsequent execution does not require any input unless there is a need to change any configuration.

In the left navigation pane, click **Configure** to provide inputs parameters and scroll to bottom of the screen to pipeline script as displayed below.

Figure 3-82 Regression - Pipeline Script



You can change parameters from **"-b"** to **"-q"** as per deployment environment and click **Save**. The content of the pipeline script is as follows:

Figure 3-83 SCP-Regression Pipeline Script

```
node ('master'){
    //a = SELECTED_NF    b = NFNAMESPACE    c = CLUSTERDOMAIN      d = DESTNAMESPACE
    //e = ATSREGISTRY   f = AUDITINTERVAL       g = GUARDTIME      h = SCPSVCNAME
    //i = SCPCONFIGSVCNAME   j = SCPNOTIFYSVCNAME   k = SCPSUBSVCNAME   l = DBSECRETNAME
    //m = MYSQLHOST n = ATSSTUBIMAGE    o = ATSSTUBCPU    p = ATSSTUBMEMORY   q = RERUN_COUNT
    sh '''
        sh /var/lib/jenkins/ocscp_tests/preTestConfig.sh \
        -a SCP \
        -b scpsvc \
        -c odyssey.lab.us.oracle.com \
        -d scpsvc \
        -e bastion-1:5000/ocats \
        -f 120 \
        -g 10 \
        -h ocscp-scp-worker \
        -i ocscp-scpc-configuration \
        -j ocscp-scpc-notification \
        -k ocscp-scpc-subscription \
        -l cred \
        -m mysql.default \
        -n ocats-gostub:1.7.0 \
        -o 0.2 \
        -p 0.1G \
        -q 0
    '''
    load "/var/lib/jenkins/ocscp_tests/jenkinsData/Jenkinsfile-Regression"
}
```

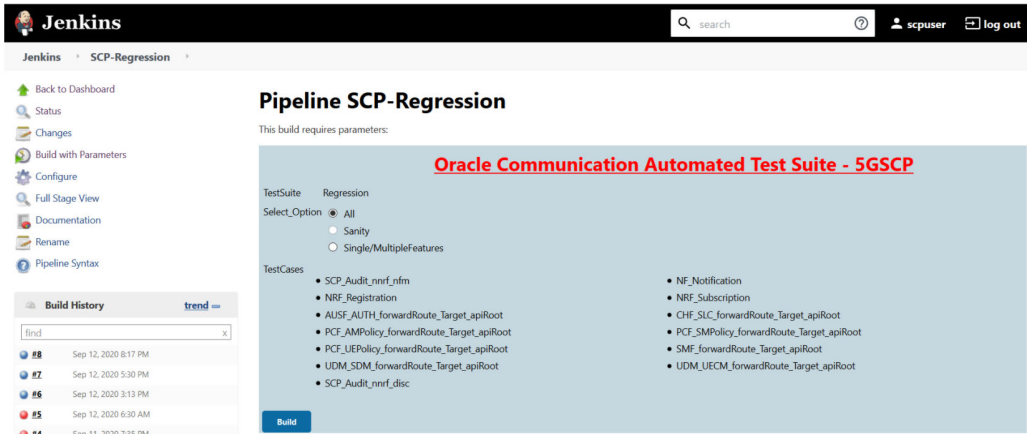
The description of parameters is as follows:

```
-a    - Selected NF
           -b    - NameSpace in which SCP is Deployed
           -c    - K8s Cluster Domain where SCP is Deployed
           -d    - Test Stubs NameSpace - Must be same as SCP Namespace
           -e    - Docker registry where test stub image is available
           -f    - Audit Interval provided in SCP Deployment file
           -g   - Guard Time provided SCP Deployment file
           -h   - SCP-Worker microservice name as provided during
deployment
           -i    - SCPC-Configuration microservice name as provided
during deployment
           -j    - SCPC-Notification microservice name as provided
during deployment
           -k   - SCPC-Subscription microservice name as provided
```

```
during deployment
          -l    - DB Secret name as provided during deployment
          -m  - Mysql Host name as provided during deployment
          -n   - Test Stub Image Name with tag
          -o   - Test Stub CPU requests and limit
          -p   - Test Stub Memory requests and limit
          -q   - re-run count
```

Click **Build with Parameters**. The following screen appears:

Figure 3-84 SCP-Regression Build with Parameters Option

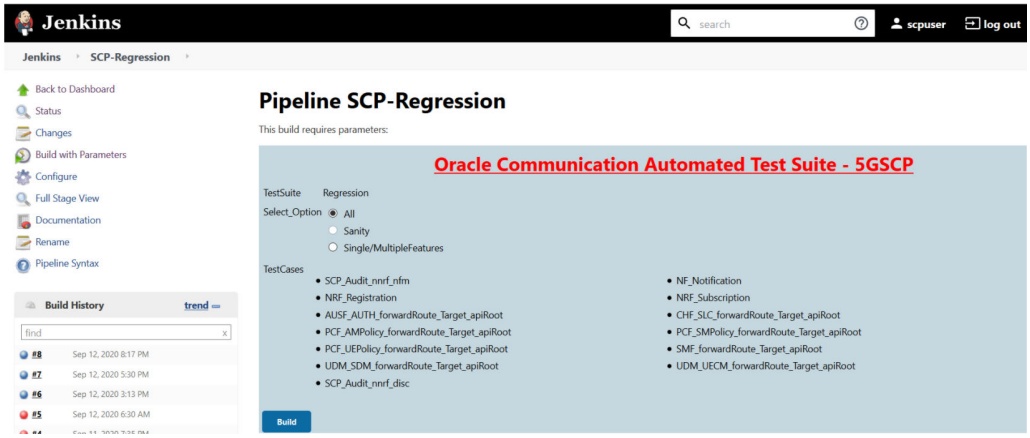


It has following three options:

- **All** - To execute all the test cases except **SCP_Audit_nnrf_disc**. If SCP is deployed with nnrf-disc for Audit or Registration with NRF is not enabled, then you should not use the All option. Instead, use Single/MultipleFeatures option to select appropriate cases for execution.
- **Sanity** - This option is not available for SCP.
- **Single/MultipleFeatures** - To execute selected test cases. You can select one or more test cases and execute using this option.

Select an appropriate option and click **Build** to start test execution.

Figure 3-85 SCP-Regression Build Option



To check execution results and logs, click the execute-tests stage of pipeline and then logs. To open execution logs console, select test execution step and double-click the execution log.

Figure 3-86 SCP-Regression Stage Logs

Executing SEPP Test Cases using ATS

To execute SEPP Test Cases using NRF ATS 1.4, you need to ensure that following prerequisites are fulfilled.

- The user must create Kubernetes secret with certificates/keys (public and private) for both plmn and n32 gateways before deploying SEPP.
- SEPP 1.4 must be deployed with default helm configurations using helm charts.
- All micro-services of SEPP should be up and running.
- The user must create Kubernetes secret with certificates/keys (public and private) for ats client and stub server microservices before deploying SEPP ATS.

- ATS is deployed using the helm charts.
- The stub is deployed using helm charts.
- Prometheus service must be up and running.
- Users can customize test cases in the custom test case folders (cust_newfeatures, cust_regression and cust_performance). They can add new test cases, remove unwanted test cases and modify existing test cases. It does not impact the original product packaged test cases available in the newfeatures, regression and performance folders. For more details, you can refer to Custom Folder Implementation.

Logging into ATS

Before logging into ATS, you need to ensure that ATS is deployed successfully using HELM charts. A sample screen is given below:

```
[root@vs-kube-master test]# helm status ocats-sepp
LAST DEPLOYED: Sat Sep 26 21:18:02 2020
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/ServiceAccount
NAME                                SECRETS  AGE
default-bddclient-serviceaccount    1        12m
default-stubserver-serviceaccount   1        12m

==> v1/Role
NAME                    AGE
default-bddclient-role   12m
default-stubserver-role  12m

==> v1beta1/RoleBinding
NAME                             AGE
default-bddclient-rolebinding    12m
default-stubserver-rolebinding   12m

==> v1/Service
NAME                  TYPE          CLUSTER-IP     EXTERNAL-IP  PORT(S)                                        AGE
ocats-sepp-bddclient  LoadBalancer  10.105.80.3    <pending>    8080:30076/TCP                                 12m
ocats-sepp-stubserver LoadBalancer  10.105.225.71  <pending>    8080:30002/TCP,8091:31242/TCP,8443:32640/TCP   12m

==> v1/Deployment
NAME                  DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
ocats-sepp-bddclient  1        1        1           1          12m
ocats-sepp-stubserver 1        1        1           1          12m

==> v1/Pod(related)
NAME                                   READY  STATUS   RESTARTS  AGE
ocats-sepp-bddclient-69b466f5fb-vk7md  1/1    Running  0         12m
ocats-sepp-stubserver-c6576dd6c-r4z2h  1/1    Running  0         12m
```

There are two ways to login to ATS Jenkins GUI.

- When an external load balancer (metalLB in case of OCCNE) is available and an external IP is provided to the ATS service, the user can login to ATS GUI using <External-IP>:8080.
- When an external IP is not provided to the ATS service, the user can open the browser and provide the external IP of the worker node and nodeport of the ATS service to login to ATS GUI.
  `<Worker-Node-IP>:<Node-Port-of-ATS>`

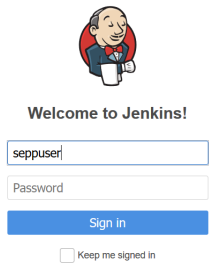
> **Note:**
>
> In the **Verifying ATS Deployment** screen, ATS nodeport is highlighted in red as 30076. For more details on ATS deployment, refer to **SEPP ATS Installation Procedure.**

Open a browser and provide IP and port details as <Worker-Node-IP>:<NodePort-of-ATS> (As per the above example: 10.98.101.171:32013). The ATS login screen appears.

Executing ATS

To execute ATS:

- Enter the username as 'seppuser' and password as 'sepppasswd'. Click **Sign in**.

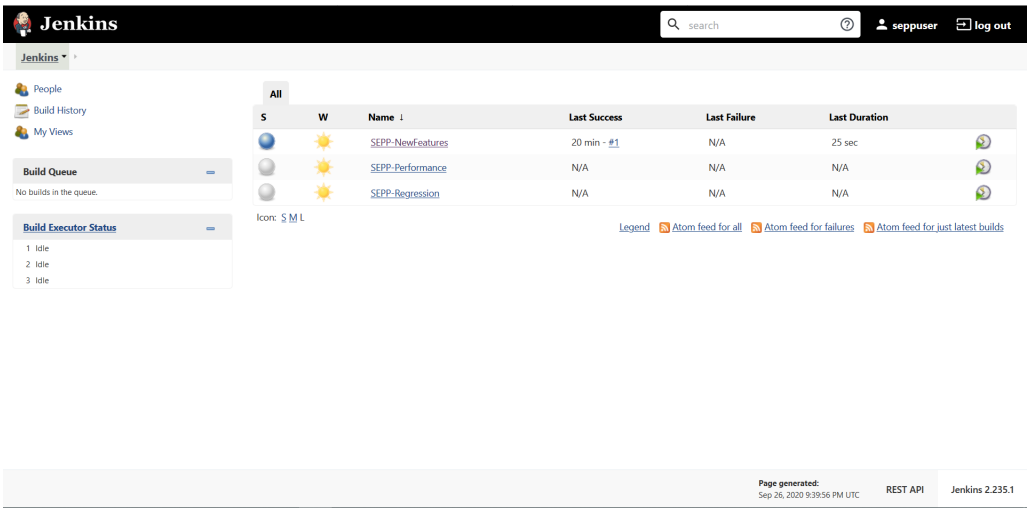


> **Note:**
>
> If you want to modify your default login password, refer to Modifying Login Password

The following screen appears showing pre-configured pipelines for SEPP individually (3 Pipelines).

- **SEPP-NewFeatures:** This pipeline has all the test cases that are delivered as part of SEPP ATS - 1.4.
- **SEPP-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
- **SEPP-Regression:** This pipleine has all the test cases of previous releases. As this is the first release of SEPP-ATS, this pipeline does not show any previous release test cases.

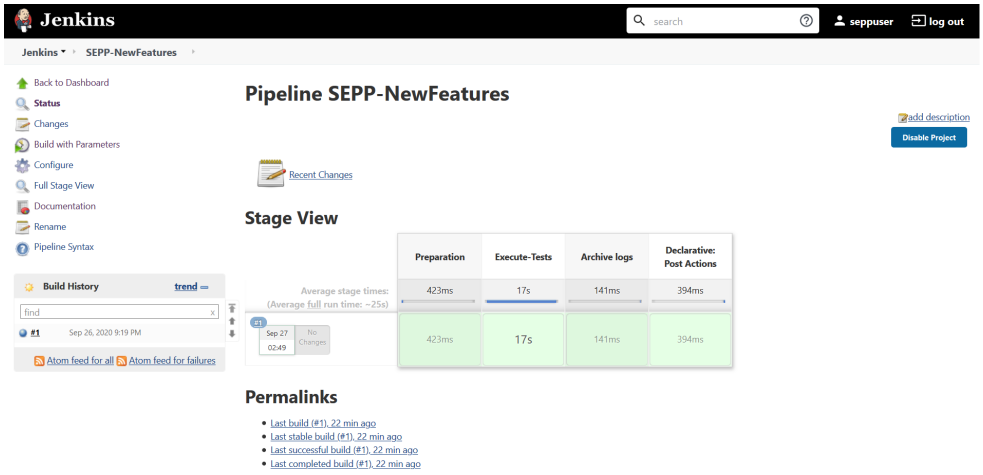
Figure 3-87 Pre-Configured Pipelines



Each one of this pipeline is explained below:

- **SEPP-NewFeatures Pipeline:** After identifying the SEPP pipelines, the user needs to do one-time configuration in ATS as per their SUT deployment. In this pipeline, all the new testcases related to SEPP are executed. To configure its parameters:
- Click **SEPP-NewFeatures** in the Name column. The following screen appears:

Figure 3-88 SEPP-NewFeatures Pipeline

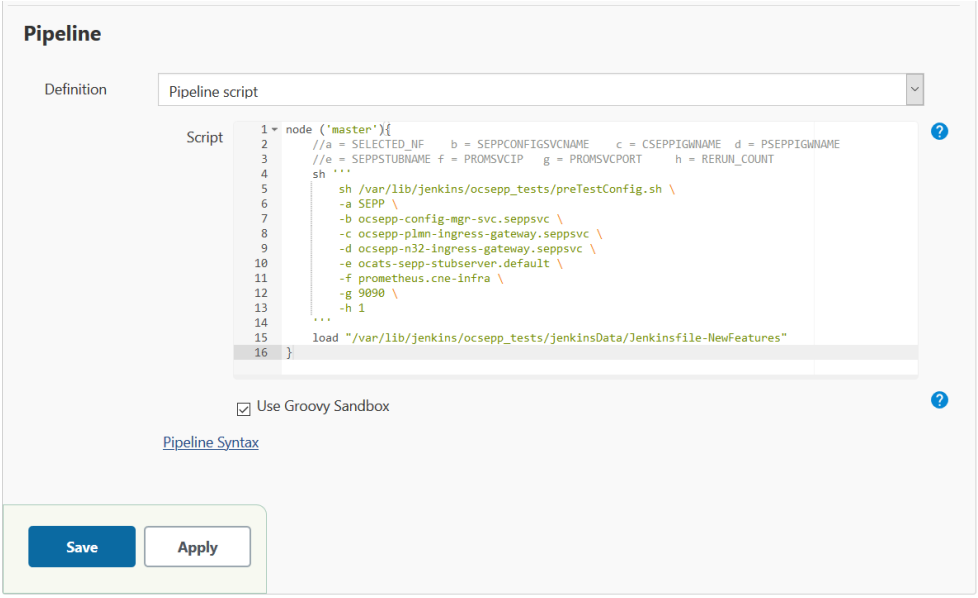


- In the above screen:
  - Click **Configure** to navigate to a screen where configuration needs to be done.
  - Click **Documentation** to view the documented test cases.
  - Click blue dots inside **Build History** box to view the success console logs of the "All" and "Sanity" respectively.

  - The **Stage View** represents already executed pipeline for the customer reference.

- Click **Configure**. Users MUST wait for the page to load completely. Once the page loads completely, click the **Pipeline** tab to reach the Pipeline configuration as shown below:

> **Note:**
>
> **MAKE SURE THAT THE SCREEN SHOWN ABOVE LOADS COMPLETELY BEFORE YOU PERFORM ANY ACTION ON IT. ALSO, DO NOT MODIFY ANY CONFIGURATION OTHER THAN DISCUSSED BELOW.**



- In the above screen, the values of the 'Pipeline script' needs to be changed. The content of the pipeline script is as follows:

```
node ('master'){
    //a = SELECTED_NF    b = SEPPCONFIGSVCNAME    c = CSEPPIGWNAME
d = PSEPPIGWNAME
    //e = SEPPSTUBNAME f = PROMSVCIP   g = PROMSVCPORT     h =
RERUN_COUNT
    sh '''
        sh /var/lib/jenkins/ocsepp_tests/preTestConfig.sh \
        -a SEPP \
        -b ocsepp-config-mgr-svc.seppsvc \
        -c ocsepp-plmn-ingress-gateway.seppsvc \
        -d ocsepp-n32-ingress-gateway.seppsvc \
        -e ocats-sepp-stubserver.default \
        -f prometheus.cne-infra \
        -g 9090 \
        -h 1
    '''
    load "/var/lib/jenkins/ocsepp_tests/jenkinsData/Jenkinsfile-
```

```
NewFeatures"
}
```

> **Note:**
>
> The User MUST NOT change any other value apart from line number 8 to line 20.

You can change only those parameters that are marked as "a" to "h" as per your requirement.

- a - Name of the NF to be tested in capital (SEPP).
- b - SEPP Config service name including namespace
- c - cSEPP Plmn Ingress gateway service name including namespace
- d - pSEPP N32 Ingress gateway service name including namespace
- e - Stub Server service name inclding namespace
- f - Prometheus service name or IP including namespace
- g - Prometheus service port
- h - Number of times the re-run of failed case is allowed (default as 2).

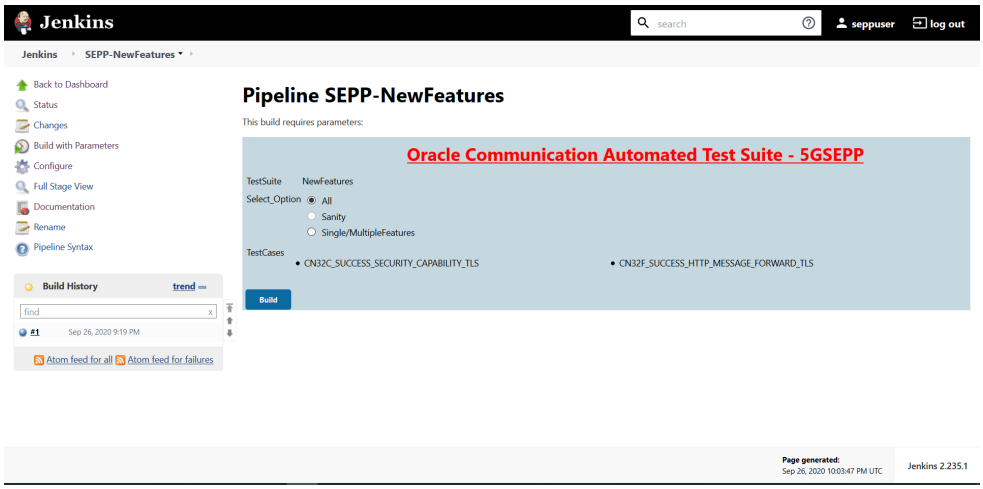
> **Note:**
>
> You do not have to change any value if OCCNE cluster is used and SEPP, ATS and STUB are deployed in ocsepp namespace.

- Click **Save** after making necessary changes. You are navigated back to the **Pipeline SEPP-NewFeatures** screen.

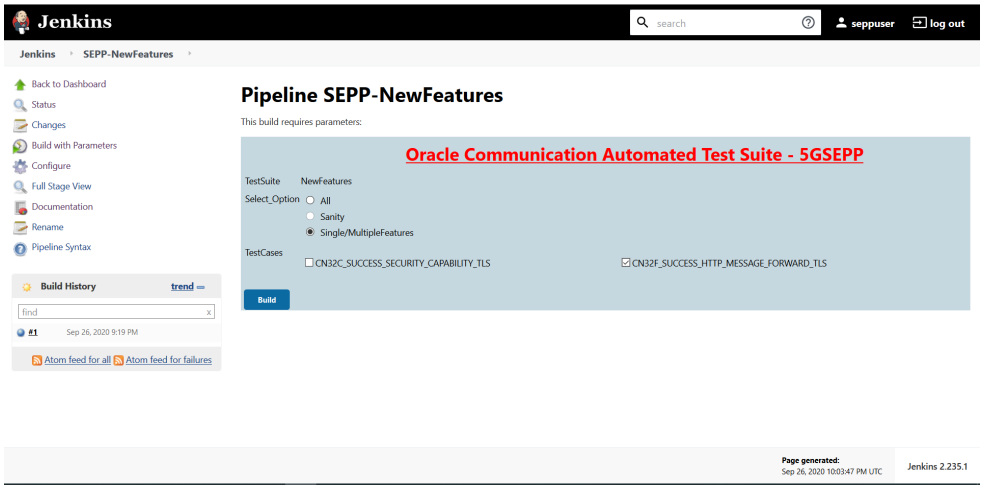
Executing SEPP Test Cases

To execute SEPP test cases:

1. Click the **Schedule a Build with parameters** icon present on the SEPP-NewFeatures screen in the extreme right column corresponding to SEPP-NewFeatures row.The following screen appears:

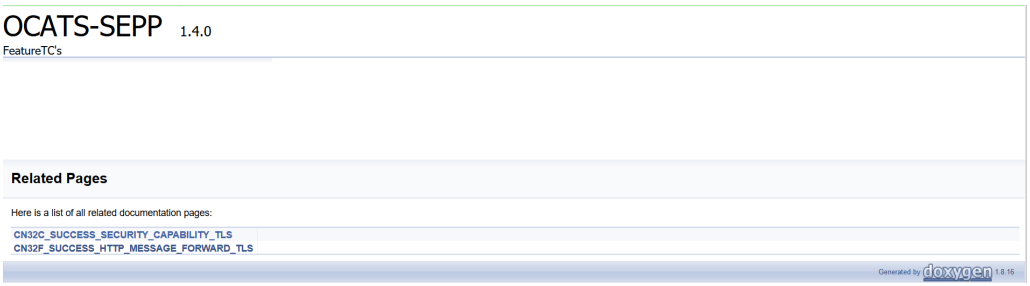


2. In the above screen, there are three **Select_Option(s)**, which are:
   - **All:** By default, all the SEPP test cases are selected for execution. User just needs to scroll down and click **Build** to execute all the test cases.
   - **Sanity**: Currently disabled.
   - **Single/MultipleFeature:** This option allows you to select any number of test cases that you want to execute from the list of total test cases available for execution. After selecting the test cases, scroll-down and click **Build**. The selected SEPP test cases are executed.

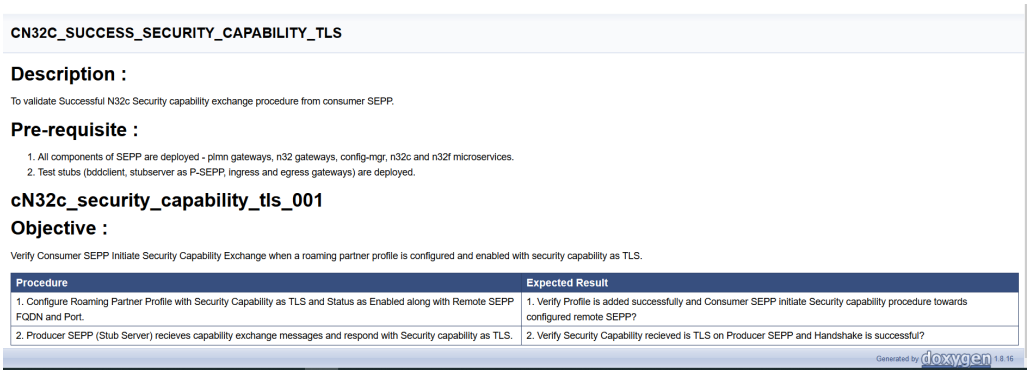


NewFeatures - Documentation

To view SEPP functionalities, go to SEPP-NewFeatures pipeline and click the **Documentation** link in the left navigation pane. The following screen appears:



User can click any functionality to view its test cases and scenarios of each test case. A sample screen is given below:



Executing SLF Test Cases using ATS

Custom Folder Implementation

Users can customize test cases in the custom test case folders (cust_newfeatures, cust_regression and cust_performance). They can add new test cases, remove unwanted test cases and modify existing test cases. It does not impact the original product packaged test cases available in the newfeatures, regression and performance folders. For more details, you can refer to Custom Folder Implementation.

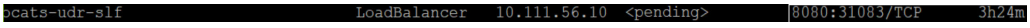
Logging into ATS

Before logging into ATS, you need to know the nodeport of the "-ocats-udr-slf" service. To get the nodeport detail, execute the following command:

```
kubectl get svc -n <slf_namespace>
```

Example: `kubectl get svc -n ocats`

Figure 3-89 SLF Nodeport



In the above screen, 31083 is the nodeport.

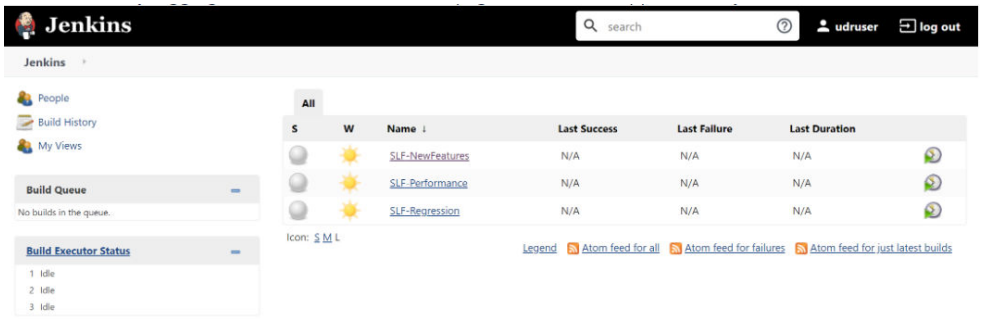
To login to ATS via Jenkins:

1. In the web browser, type **http://<Worker IP>:<port obtained above>** and press **Enter**.
   Example: http://10.75.225.49:31083

   The Login screen appears.
2. Enter the username as 'udruser' and password as 'udrpasswd'. Click **Sign in**. A screen with pre-configured pipelines for SLF appears (3 pipelines).
   - **SLF-New-Features:** This pipeline has all the test cases, which are delivered as part of SLF ATS - 1.8.0.
   - **SLF-Performance:** This pipeline is not operational as of now. It is reserved for future releases of ATS.
   - **SLF-Regression:** This pipeline has all the test cases of previous releases.

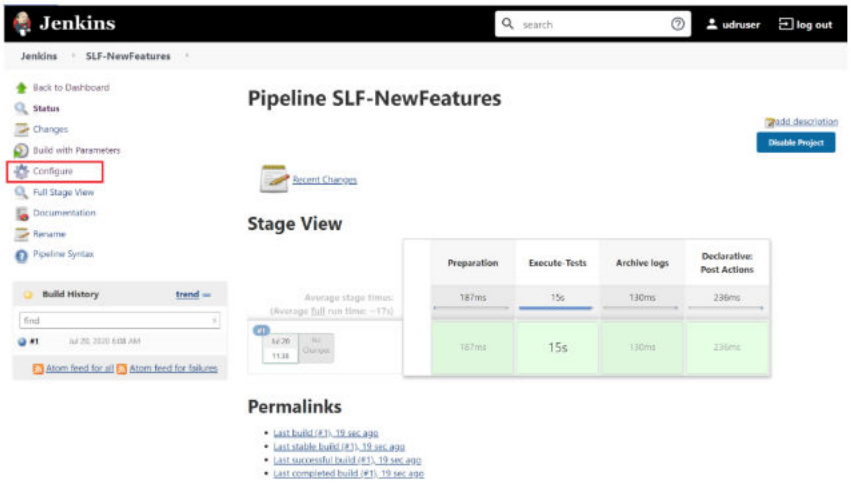
> **Note:**
>
> If you want to modify your default login password, refer to Modifying Login Password

Figure 3-90 SLF Pre-configured Pipelines



3. Click **SLF-NewFeatures**. The following screen appears:

Figure 3-91 SLF-NewFeatures Configure



4. Click **Configure** in the left navigation pane. The **General** tab appears. User **MUST** wait for the page to load completely.
5. Once the page loads completely, click the **Advanced Project Options** tab. Scroll down to reach the **Pipeline** configuration as shown below:
MAKE SURE THAT THE SCREEN SHOWN BELOW LOADS COMPLETELY BEFORE YOU PERFORM ANY ACTION ON IT. ALSO, DO NOT MODIFY ANY CONFIGURATION OTHER THAN DISCUSSED BELOW.

Figure 3-92 SLF Configuration Parameters - Pipeline Tab

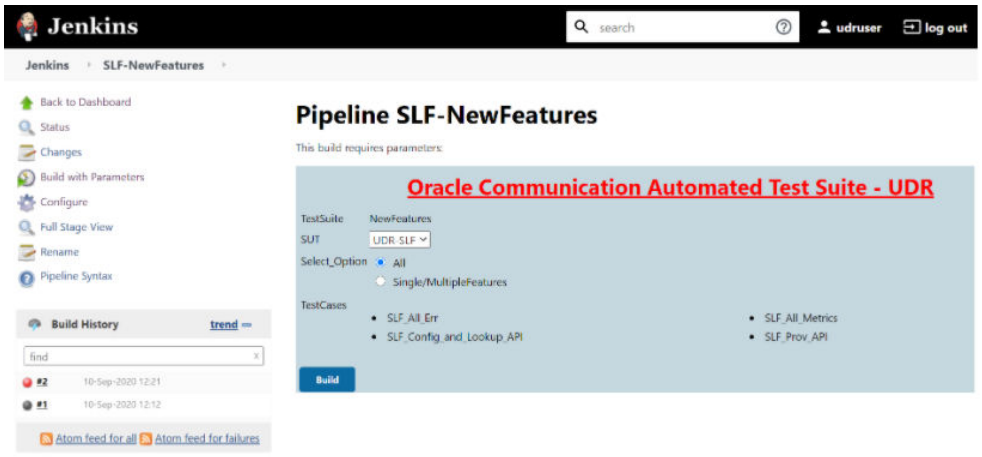
You **SHOULD NOT** change any other value apart from **line number 36 to line 58**. It means the parameters marked as **"a"** - to - **"w"** can only be changed as per user requirement. The detail about these parameters are provided as comments in line number 6 to 12. The parameters description is as follows:

> **Note:**
>
> The parameters that you should modify to execute UDR-SLF ATS SUT are: a, b, d, e, n, o and p.

- **a** - Name of the NF to be tested in capital (SLF).
- **b** - Namespace in which the udr is deployed.
- **c** - Namespace in which ProvGw is Deployed
- **d** - Name of UDR1_ingressgateway_service.namespace preferred in segment 1(seg1ocudr1-ingressgateway.ocudr).
- **e** - Port of UDR1 ingressgateway service (80)
- **f** - Name of UDR2_ingressgateway_service.namespace in segment 1(seg1ocudr2-ingressgateway.ocudr)
- **g** - Port of UDR2 ingressgateway service (80)
- **h** - Name of UDR3_ingressgateway_service.namespace preferred in segment 2(seg2ocudr1-ingressgateway.ocudr)
- **i** - Port of UDR3 ingressgateway service (80)
- **j** - Name of UDR4_ingressgateway_service.namespace in segment 2(seg2ocudr2-ingressgateway.ocudr)
- **k** - Port of UDR4 ingressgateway service (80)
- **l** - Name of PROVGW_ingressgateway_service.namespace (provgw-prov-ingressgateway.ocudr)
- **m**- Port of PROVGW ingressgateway service (80)
- **n** - Name_of_Prometheus_service.namespace (occne-prometheus-server.occne-infra)
- **o** - Port of Prometheus service (80)
- **p** - Number of times the re-run of failed case is allowed (default as 2)
- **q** - Name of Kubernetes Host server (kubernetes.default)
- **r** - Port of Kubernetes Host server (80)
- **s** - Mode of Communication between Prov-gateway and UDR-SLF (Can be either IP or fqdn)
- **t** - Helm Name for UDR1 (seg1ocudr1)
- **u** - Helm Name for UDR2 (seg1ocudr2)
- **v** - Helm Name for UDR3 (seg2ocudr1)
- **w**- Helm Name for UDR4 (seg2ocudr2)

6. Click **Save** after making neccesary changes. The SLF-NewFeatures screen appears.
7. Click **Build with Parameters**. The following screen appears:

Figure 3-93 SLF Build with Parameters



Note:

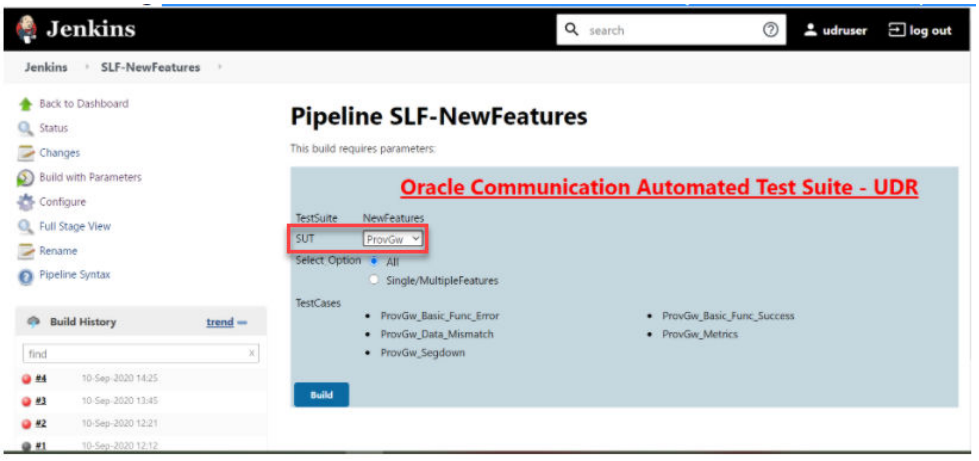
To execute UDR-SLF related test cases, select UDR-SLF from the **SUT** list. This is the default option.

To execute ProvGw related test cases, select ProvGw from the SUT list.

Note:

To execute the ProvGw test cases, it is mandatory to have 4 UDRs up and running in the same namespace as that of UDR-SLF-ATS.

Figure 3-94 SUT as ProvGw



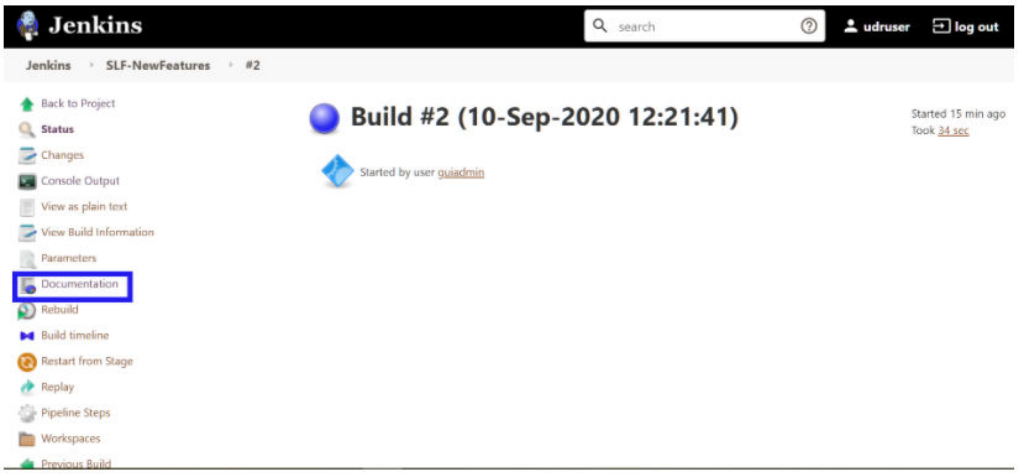
In the above screen, there are two **Select_Option**(s), which are:

- **All:** By default, all the SLF test cases are selected for execution. User just need to scroll down and click **Build** to execute all the test cases.
- **Single/MultipleFeatures:** This option allows you to select any number of test cases that you want to execute from the list of total test cases available for execution. After selecting the test cases, scroll-down and click **Build**. The selected SLF test cases are executed.

NewFeatures-Documentation

To view the SLF functionalities, click **Documentation** link in the left navigation pane (present inside the build), as shown below:

Figure 3-95 SLF-NewFeatures Documentation Option



The following screen appears:

Figure 3-96 SLF-NewFeatures Documentation

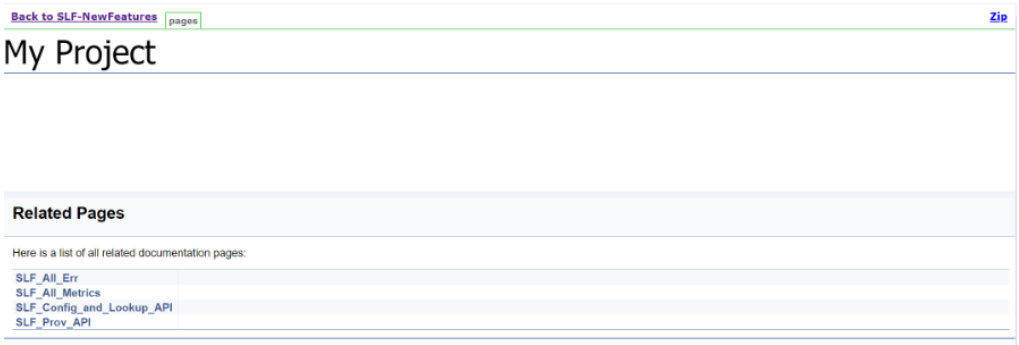
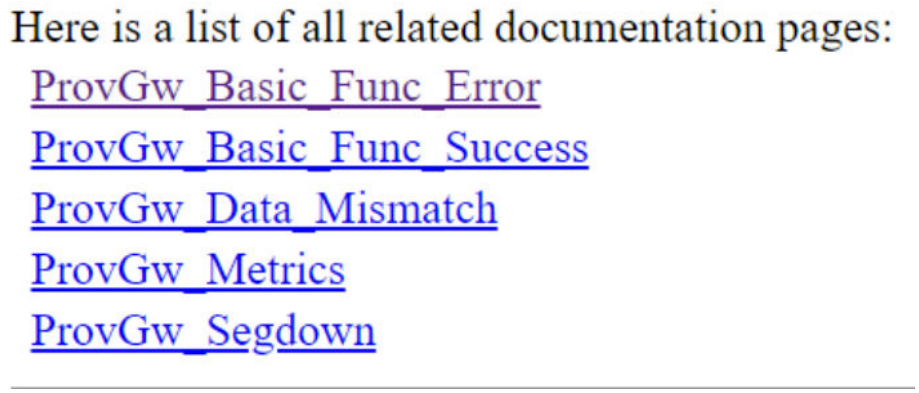


Figure 3-97 ProvGw-NewFeatures Documentation



> **Note:**
>
> Documentation option appears only if New-Features pipeline is executed atleast once.

You can click any functionality to view its test cases and scenarios for each test case. For example, on click of SLF_All_Err, following screen appears:

Figure 3-98 Sample: SLF Test Case Description

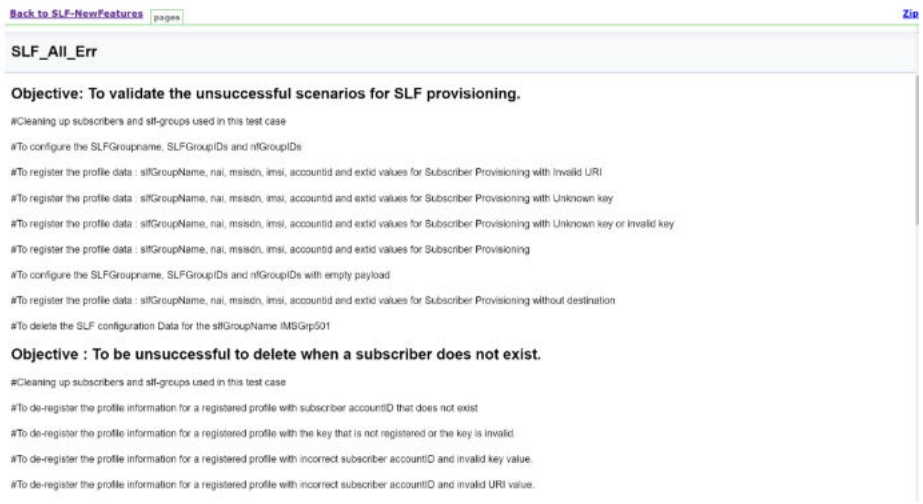
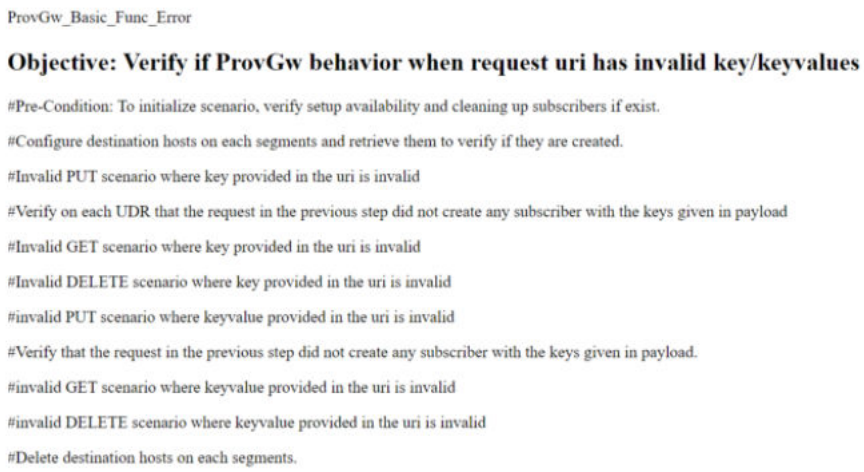


Figure 3-99 Sample: ProvGw Test Case Description


ProvGw_Basic_Func_Error

Objective: Verify if ProvGw behavior when request uri has invalid key/keyvalues

#Pre-Condition: To initialize scenario, verify setup availability and cleaning up subscribers if exist.

#Configure destination hosts on each segments and retrieve them to verify if they are created.

#Invalid PUT scenario where key provided in the uri is invalid

#Verify on each UDR that the request in the previous step did not create any subscriber with the keys given in payload

#Invalid GET scenario where key provided in the uri is invalid

#Invalid DELETE scenario where key provided in the uri is invalid

#invalid PUT scenario where keyvalue provided in the uri is invalid

#Verify that the request in the previous step did not create any subscriber with the keys given in payload.

#invalid GET scenario where keyvalue provided in the uri is invalid

#invalid DELETE scenario where keyvalue provided in the uri is invalid

#Delete destination hosts on each segments.

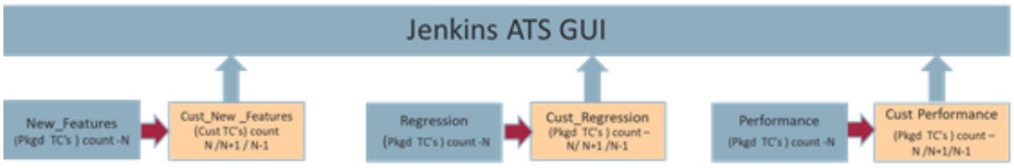
Based on the functionalities covered under Documentation, the **Build Requires Parameters** screen displays test cases. To navigate back to the Pipeline SLF-NewFeatures screen, click **Back to SLF-NewFeatures** link available on top left corner of the screen.

A
Custom Folder Implementation

New custom test cases folders (**cust_newfeatures**, **cust_regression** and **cust_performance**) have been introduced to accommodate the customization's to original product packaged test cases. These folders carry the customized test cases (any new test cases added by customers / subset of test cases from the original product supplied test cases / modified test cases).

Initially when packaged and released, both the product test case folders (newfeatures, regression and performance) and the custom test case folders (cust_newfeatures, cust_regression and cust_performance) carries same set of test cases. Subsequently, customers can use the custom test case folders to carry out any customization's from their side (updates / additions / deletions of test cases) without disturbing the original product packaged test cases / folders. Jenkins always pick the test cases from the custom test cases folders.

Figure A-1 Summary of Custom Folder Implementation



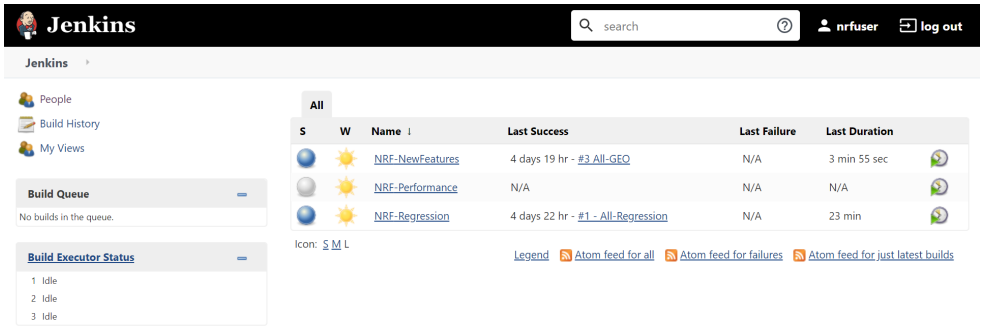
B
Modifying Login Password

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

If the user wants to modify its login password, the ATS application allows to do so. To modify login password:

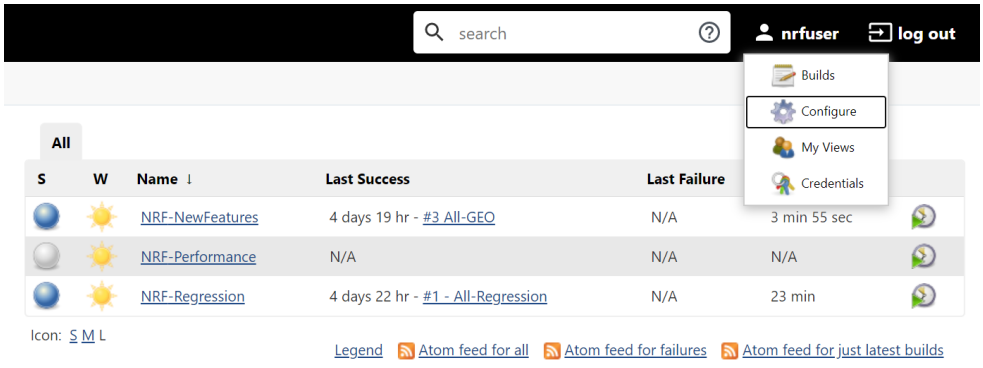
1. Login to ATS application using default login credentials. The home screen of respective NF appears showing its pre-configured pipelines.

Figure B-1 Sample: NRF Home Screen



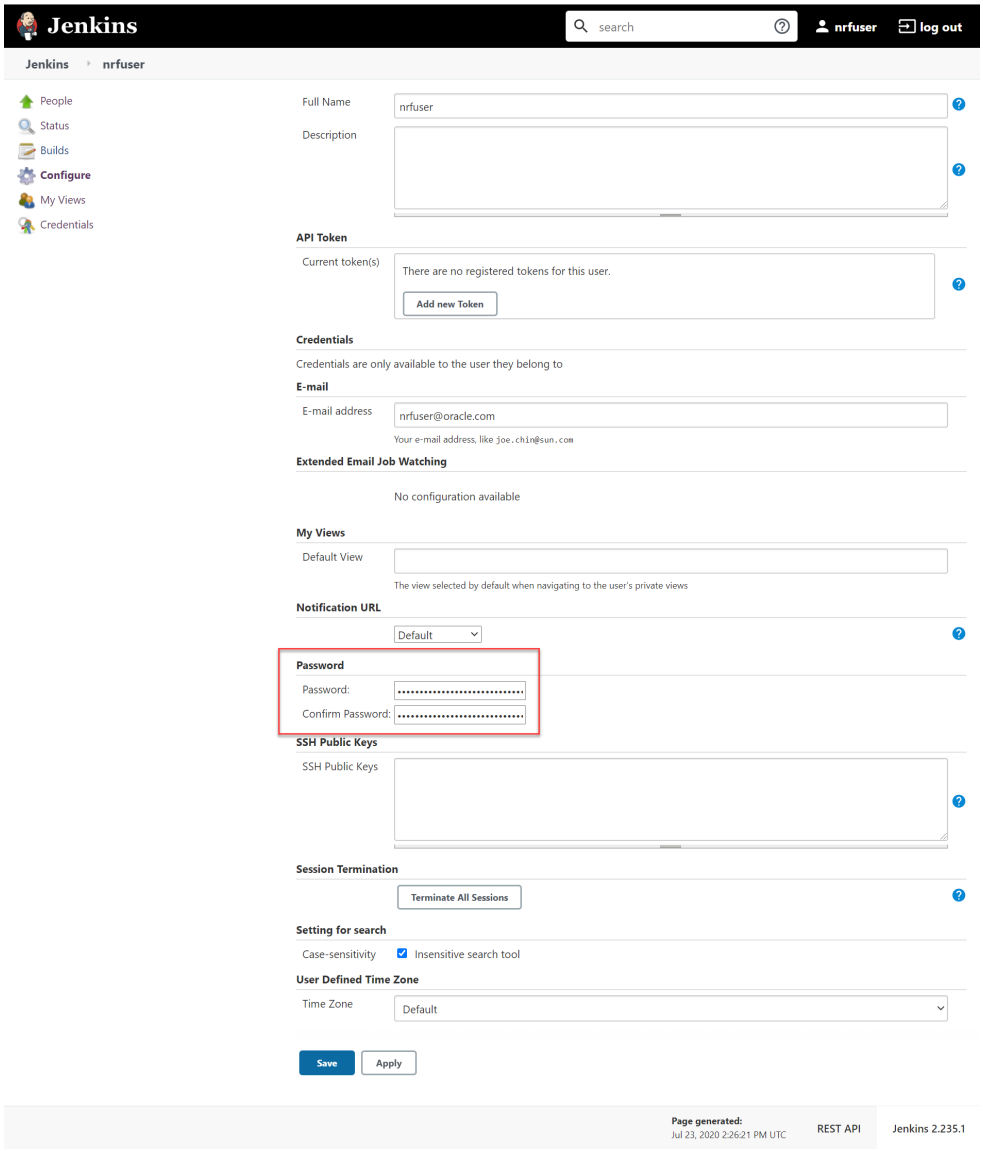
2. Hover-over logged-in user name and click the down arrow. Click **Configure** as shown below.

Figure B-2 Configure Option



3. The following screen appears.

Figure B-3 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.