ORACLE

Technical brief: Securely Accessing ExaCS or DBCS with Bastion Service

A example guide on how Bastion service can improve the security and governance of Linux VM access to Database Cloud Service or ExadataCloud Service

Inge Os,
Master Principal Cloud Specialist
Oracle EMEA

October, 2022, Version 1.0

Purpose

In the shared responsibility model of PaaS Services like Database Cloud Service, DBCS, or Exadata Cloud Services, ExaCS, the service consumers are granted OS access, and have some responsibilities requiring access to the Linux OS, as user *oracle* or with group membership equal to the *oracle* user.

The services is created with the *opc* account, authenticated with a private/public ssh keypair. User *opc* is configured to offer sudo *root* access.

To improve the security for DBCS and ExaCS, customers would like to limit operator access to *opc* and *root* accounts, and enforce an access model to the Linux VM, or nodes by:

- Prevent any logon to *opc* or *root* accounts
- Keep full central governance over which operator is granted access rights, through the centralized IAM Domain in Oracle OCI
- Generate centralized audit records of any logons to the Linux VM/nodes

From a governance perspective, organizations are required to enforce the usage of named users, and not common accounts Linux like *opc* or *oracle*, for the purpose of accountability and auditability

Central governance, with central user CRUD, is the preferred opting, avoiding fragmented user management

The purpose of this document is to give the reader inspiration to how such a model may be implemented with Oracle OCI Bastion services.

Overview of the OCI Bastion Service

Oracle Cloud Infrastructure (OCI) Bastion service, is a fully managed service providing secure and ephemeral Secure Shell (SSH) access to the private resources in OCI. OCI Bastion service, like the bastion fortress of medieval times, improves security posture by providing an additional layer of defence against external threats.

The cost, deployment, and management of using a jump host is a huge pain point which can lead to a weaker security posture for the customer. The OCI Bastion service removes the public and private virtual cloud networking (VCN) hassle for access to a jump host. No public IP is needed, resulting in no surface attack area or zero-day vulnerabilities with a dedicated jump host. Customers also eliminate shared credentials, broad access limits, and other bad habits of using jump hosts. OCI Bastion service integrates with OCI Identity and Access Management (IAM) and allows the organization to control who can access a bastion or a session and what they can do with those resources.

The main security features of the OCI Bastion service are:

- Provides restricted and time-limited secure access to resources that don't have public endpoints and require strict resource access controls

- With Oracle Cloud Infrastructure (OCI) Bastion service, customers can enable access to private hosts without deploying and maintaining a jump host.
- Gaining improved security posture with identity-based permissions and a centralized, audited, and time-bound SSH session
- Oracle OCI Bastion service utilizes Oracle OCI IAM, removing the need for user governance on Linux host, centralizing all user governance, RBAC access to one single instance of Oracle OCI IAM

For more info, please refer to:

- Oracle OCI Bastion services

https://docs.oracle.com/en-us/iaas/Content/Bastion/Concepts/bastionoverview.htm

- Oracle OCI Identity Domains

https://docs.oracle.com/en-us/iaas/Content/Identity/getstarted/identity-domains.htm

Traditional Linux user governance

This section outlines traditional, Linux level, approaches to governance of VM and nodes.

The users Linux account is assigned to the required groups at Linux level to provide access to the oracle software owner and grid software owner.

DBCS and ExaCS runs on Oracle Enterprise Linux, OEL, supporting standard Linux user management/user governance

From a Linux perspective the following will partly support the common requirements:

Oracle Linux account is shared:

- Don't share the *opc* private key, no one is allowed to use ssh with the *opc* private key.
- Each user signing into the Linux node has their own private key. The *oracle* account is public and Shared *oracle* account
- Each user signing into the Linux node has their own private key, but signs into user *oracle*. The *oracle* account is public and is shared. The user's public keys are stored in ~/.ssh/authorized_keys

Individual accounts

- For each user a personal account is created with the user's public key in the users authorized keys file, or the user is authenticated via LDAP over Linux PAM
- The users Linux account is assigned to the required groups at Linux level to provide access to the *oracle* software owner and grid software owner

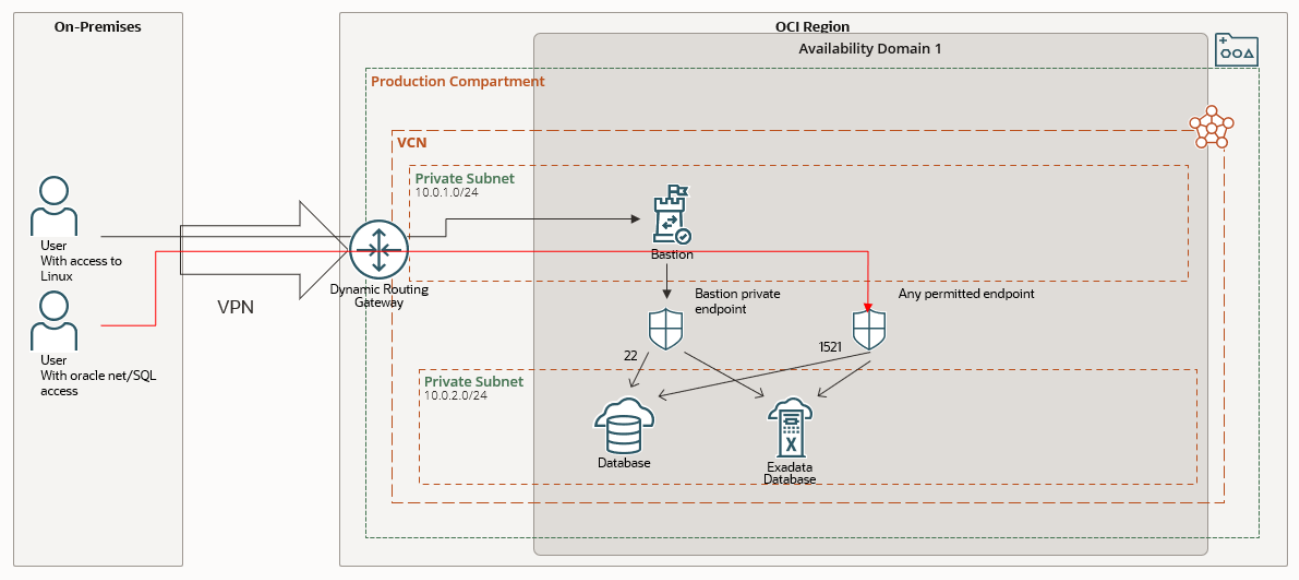
The traditional Linux governance or Linux type of bastion/jump server has a few drawbacks like:

- No central user governance, users or keys needs to be maintained at VM or node level, at all VMs and nodes
- Central IAM may be deployed with agents and IAM integration, yet another IAM solution to be maintained
- No central audit for malicious user access detection

Use Oracle OCI Bastion Service

With the usage of the Bastion service, an ephemeral ssh tunnel is created for a limited period of time. The only permitted traffic to the VM or compute node is routed though the Bastion service.

At network level, the Network Security Group, NSG, for the VM or compute node is configured, only to allow traffic over port 22 from the endpoint of the Bastion service.



Simplified network example network configuration

Security Rules

These security rules apply to all VNICs in this network security group. You can filter the list by ingress or egress. There can be other security rules that apply to a given VNIC in this group: from any other network security groups the VNIC is in, and any security lists associated with the VNIC's subnet. Learn more about security rules.

Add Rules | Edit | Remove

Direction	Source or Destination	Protocol	Details	Description
Direction: Egress **Stateless:** No	**Destination Type:** CIDR **Destination:** 10.3.0.0/16	All Protocols	**Allow:** All tra... Show	Any outbound traffic to local sub net
Direction: Ingress **Stateless:** No	**Source Type:** CIDR **Source:** 10.3.4.100/32	TCP	**Source Port Range:** All **Destination Port Range:** 22 **Allow:** TCP tra... Show	Only accept ssh access from ba stion service

Example of NSG rule protecting the DBCS node.

All access to DBCS or ExaCS Linux VM is tunnelled though the Bastion service

The Bastion service resides in a compartment, and only OCI users with access, though OCI policies, to the compartment and the Bastion service will be able to create a bastion ssh session to the Linux VM.

The Bastion service is configured with a private/public ssh key pair, with the public key in the authorized_keys file of the *oracle* user. The *opc* account private key is not used and not shared.

The VCN network is configured so the only permittable access over port 22 is from the private endpoint of the bastion host. Oracle net/SQL traffic over port 1521/1522 is permitted from the private subnet

Bastion service support two types of ssh tunnel:

- Managed SSH Session, for compute nodes with the OCI agent running
- SSH Port forwarding session, for compute nodes, like DBCS and ExaCS, without running OCI agent.

For the protection of DBCS and ExaCS VMs/nodes the SSH Port Forwarding session type is required.

Bastion service configuration

Create the Bastion service, which created the private endpoint in the subnet

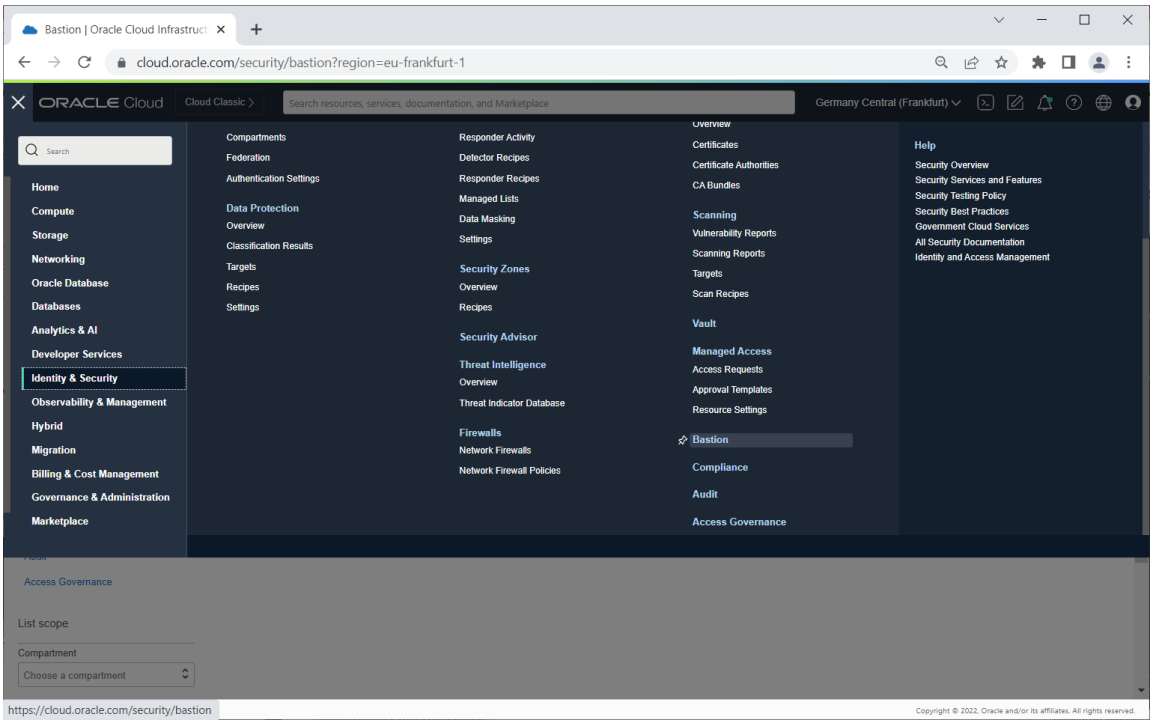
Create a private ssh key pair pr. user that needs access. Upload the public key to ~/.ssh/authorized_keys in the *oracle* account on the DBCS or ExaCS server

For each time access is needed, create a Bastion session. The Bastion session is valid for 3 hours, The bastion session generates a new ssh private/public key pair, used for ssh connection to the Bastion service

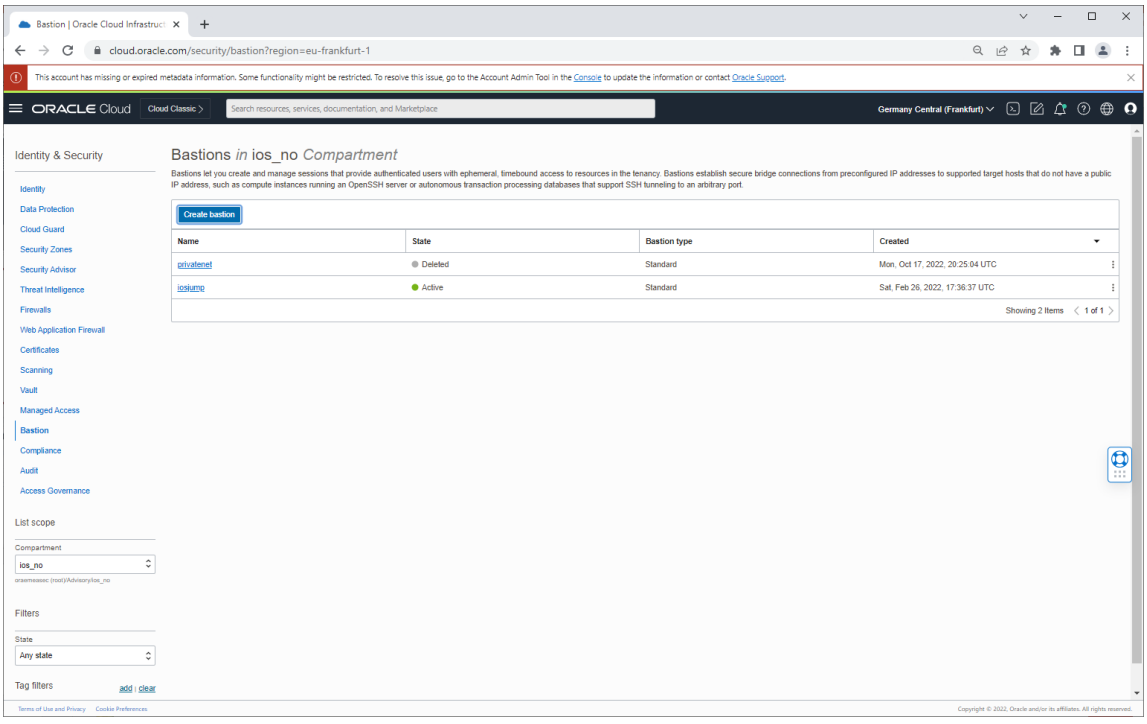
The private key for authentication of the *oracle* Linux account is tunnelled through the Bastion session

Bastion service creation

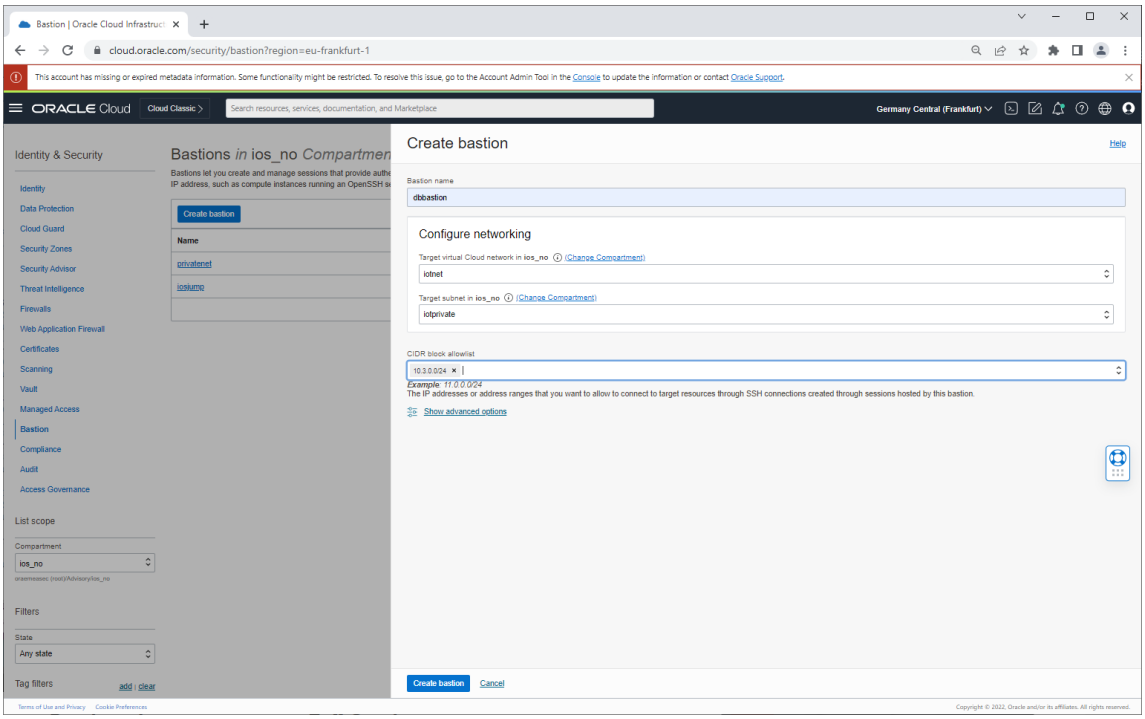
This section is a walk through of creation of the Bastion service, a prerequisite for creating any Bastion session



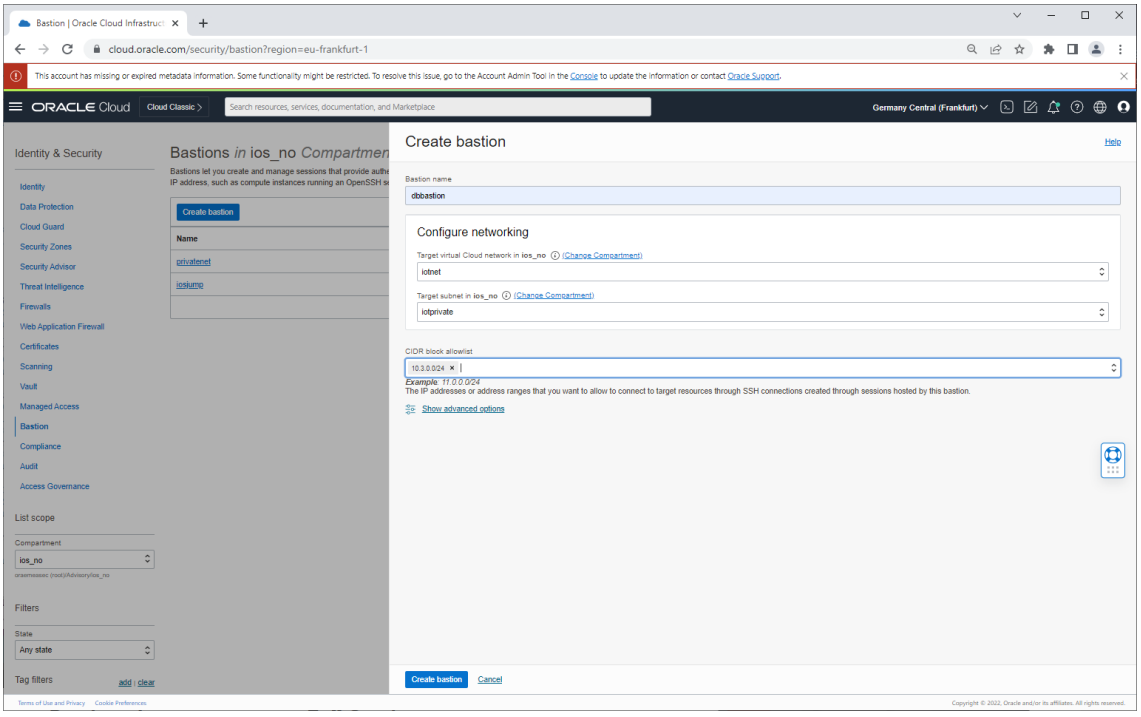
Navigate to the bastion item under Identity & Security



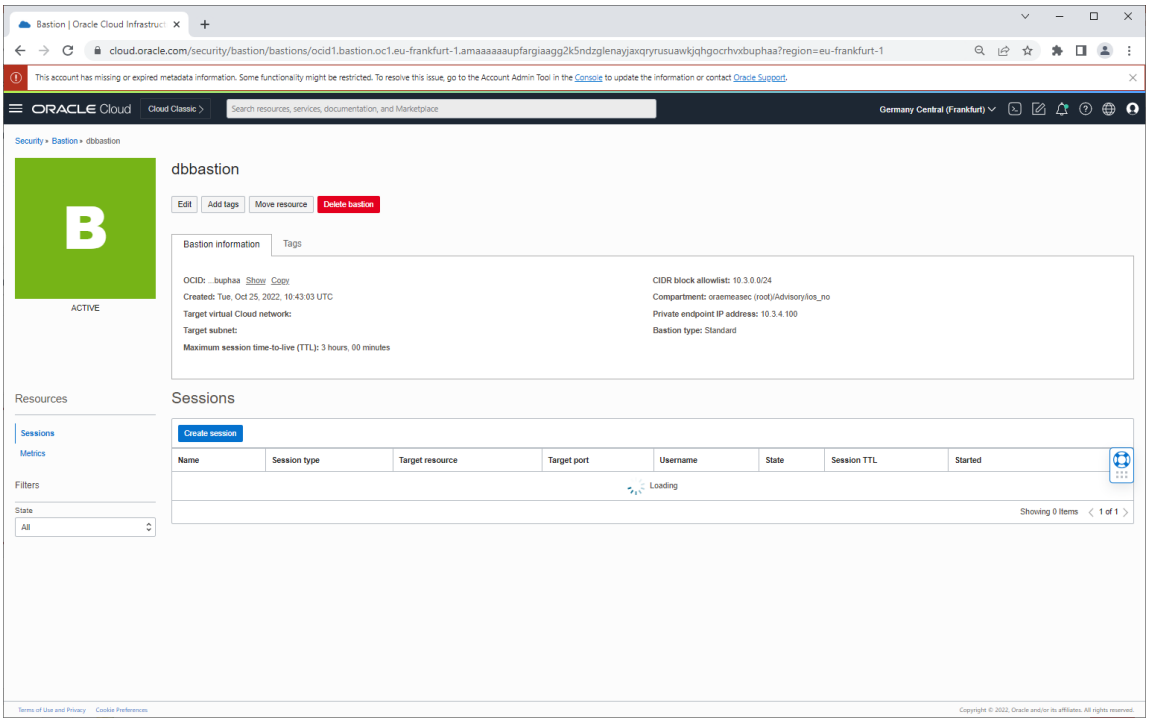
Select target compartment for the Bastion service.



Define which subnet the bastion will reside in. this defines the private endpoint, configured in the NSG later.



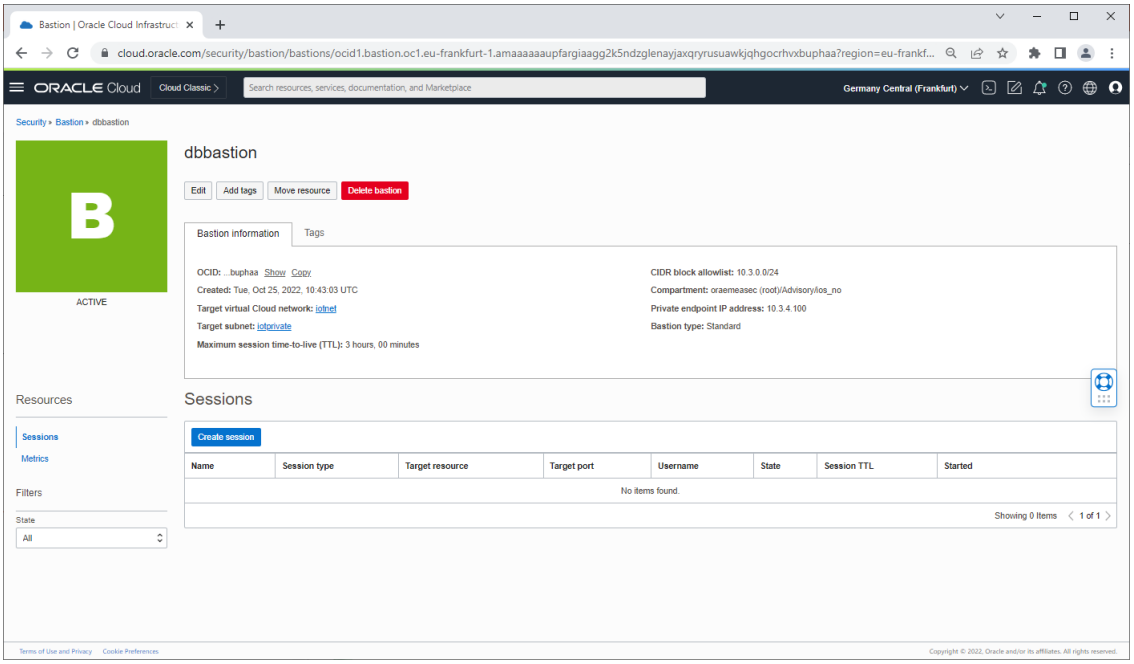
Configure allowable inbound CDIR range. Typically, the CDIR range of the VPN connection will be used.



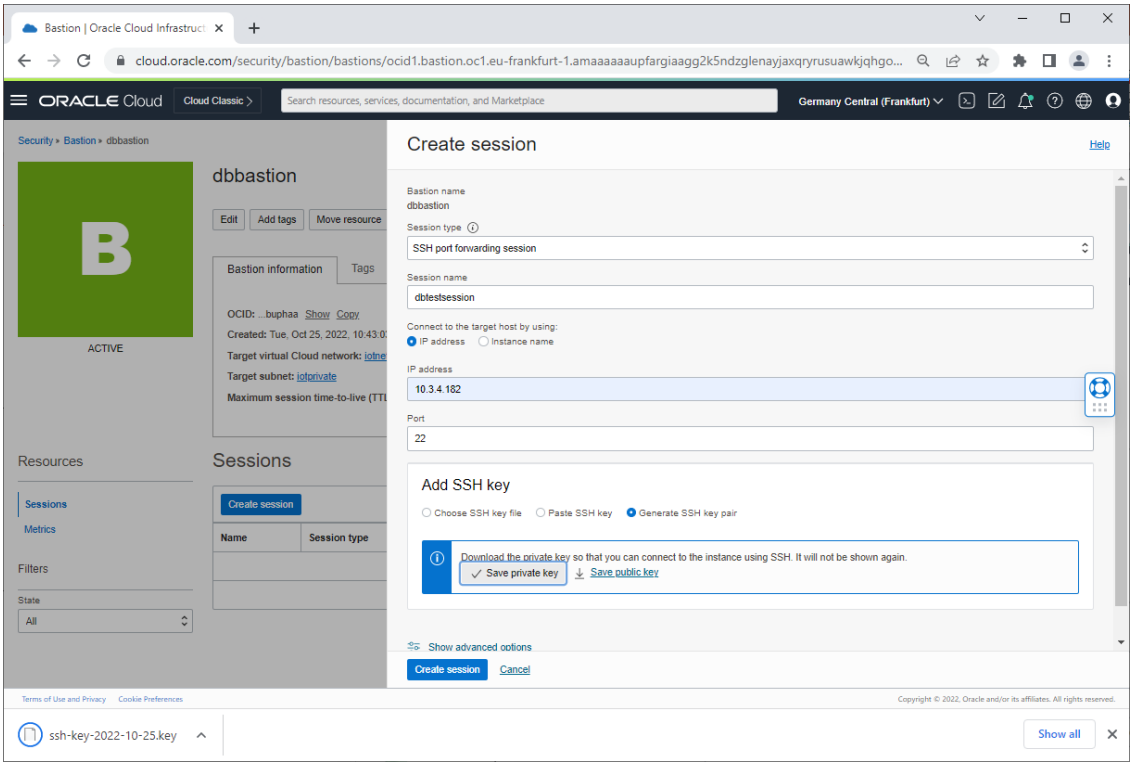
The Bastion service is created, take a not of the private endpoint, for later NSG configuration

Bastion session creation

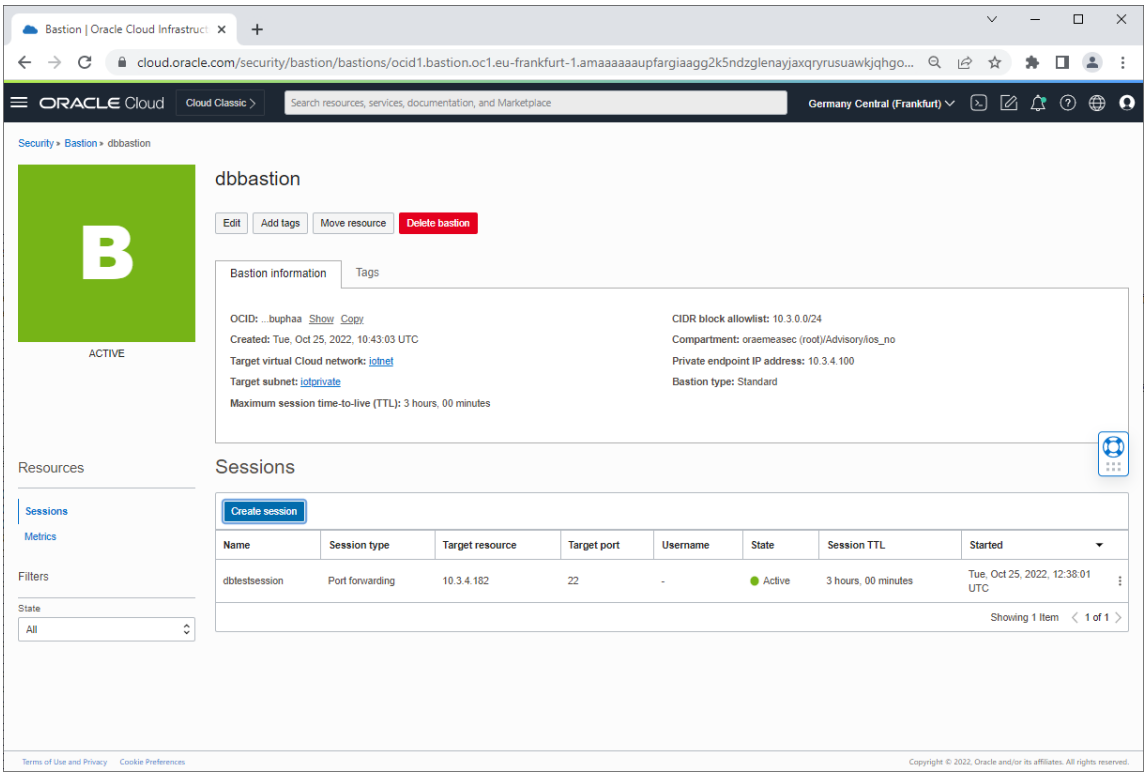
A bastion session is a time limited access granted to a user, that is audited and managed by OCI IAM policies.



Navigate to the applicable Bastion service



Select SSH Port Forwarding, generate ssh key pair and save the private key. The private key is later uploaded to the ssh agent



The session is created with a default expire time of 180 min.

Traffic flow

The DBCS or ExaCS instances resided in a separate subnet, 10.0.2.0/24, and Network Security Groups, NSG, is configured for permitting ingress/egress traffic.

With a NSG, the only permitted traffic over port 22 to the DBCS instance or ExaCS instance s from the private endpoint of the Bastion service, resides in the 10.0.1.0/24 subnet

Oracle Net traffic is configured based on the application and oracle net/SQL traffic requirements, in the example, only from the 10.0.1.0/24 subnet

The typical NSG firewall configuration for the target VM/node will allow for traffic on port 22 only from the Bastion service private endpoint, as an example 10.2.4.100/32, and open for port 1521/1522 for applicable database traffic.

ssh access to target resource via ssh tunnel

1) Start the ssh-agent

   ```
   eval `ssh-agent`
   ```

2) Upload the users own private key

   ```
   ssh-add myprivatekey.key
   ```

3) Upload the private key created by the bastion session

   ```
   ssh-add bastion-session.key
   ```

4) Create a resident ssh tunnel to the target, through the Bastion service

   ```
   ssh -N -4 -L 8222:10.3.4.182:22 -p 22 ocid1.bastionsession.oc1.eu-frankfurt-1.a*****@host.bastion.eu-frankfurt-1.oci.oraclecloud.com
   ```

5) Connect to the target over the ssh tunnel

   ```
   ssh oracle@localhost -p 8222
   ```

```
[ios@ios3 ~]$ ssh-add -l
The agent has no identities.
[ios@ios3 ~]$ ssh-add .ssh/iosstd
Identity added: .ssh/iosstd (.ssh/iosstd)
[ios@ios3 ~]$ ssh-add .ssh/ses.key
Identity added: .ssh/ses.key (.ssh/ses.key)
[ios@ios3 ~]$ ssh-add -l
2048 SHA256:zoFfJmFc0GzGZR/7j8tG/+8fswxLnUR0IvTWB9LvtaY .ssh/iosstd (RSA)
2048 SHA256:qS+sUTND8xnghyB2vFsuZ2ggc79juxOXfRTjB9nYtqI .ssh/ses.key (RSA)
[ios@ios3 ~]$ ssh -4 -N -L 8222:10.3.4.182:22 -p 22 ocid1.bastionsession.oc1.eu-frankfurt-1.amaaaaaaupfargia
jmymczp2rbe6qxthyg24jg4klyanpmyl77q457xhw33q@host.bastion.eu-frankfurt-1.oci.oraclecloud.com
```

Example of staring the ssh agent

```
[ios@ios3 cloud_scripts]$
[ios@ios3 cloud_scripts]$
[ios@ios3 cloud_scripts]$
[ios@ios3 cloud_scripts]$ ssh oracle@localhost -p 8222
Last login: Tue Oct 25 12:51:07 2022
[oracle@ios ~]$ uname -a
Linux ios 4.14.35-2047.510.5.5.el7uek.x86_64 #2 SMP Fri Jan 28 08:33:42 PST 2022 x86_64 x86_64 x86_64 GNU/Linux
[oracle@ios ~]$ exit
logout
Connection to localhost closed.
[ios@ios3 cloud_scripts]$
```

Example Session

Example script for creating Bastion service session, Managed SSH Session with OCI CLI

The code below is inspirational, how to build a script using OCI CLI to create a type of bastion session. For the script to be runnable, you need to create a OCI API Key and configure the OCI CLI.

Please review:

```
#!/bin/bash
#
# Example script that creates a managed ssh session with the bastion service
#
PYENV=/home/ios/py38
OCIPROFILE=oraemeasec
BASTIONOSID="ocid1.bastion.oc1.eu-frankfurt-
1.amaaaaaaupfargial6ibktqk7xmz6kywqkcwsxrmqrdqjsyyf24qvwcqnbpq"
SSHKEYDIR="/home/ios/.ssh/"
SSHPUBFILE="bastion_pub"
SSHPRIVATEFILE="bastion.pem"
DISPLAYNAME="iosjumpv3-session"
RESOURCEID="ocid1.instance.oc1.eu-frankfurt-
1.antheljsupfargic47fjgar7f34zdiccsv5wmrfsv4bnkfnyindlh6ew46ka"
OSUSERNAME=opc
RESOURCEPORT=22
RESOURCEIPADDRESS="10.3.4.86"
OCIREGION="eu-frankfurt-1"
TIMETOLIVE=1800
TEMPFILE=/home/ios/tempCreateSession.json
#
# Activate python env for OCI config
#
source py38env/bin/activate
#
# build inputfile for session creation
#
echo '{
  "bastionId": "'${BASTIONOSID}'",
  "displayName": "'${DISPLAYNAME}'",
  "keyType": "PUB",
  "maxWaitSeconds": 0,
  "sessionTtl": "string",
  "session-ttl-in-seconds": '${TIMETOLIVE}',
  "sshPublicKeyFile": "'${SSHKEYDIR}${SSHPUBFILE}'",
   "target-resource-details": {
        "session-type": "MANAGED_SSH",
        "target-resource-id": "'${RESOURCEID}'",
        "target-resource-operating-system-user-name": "'${OSUSERNAME}'",
        "target-resource-port": '${RESOURCEPORT}',
        "target-resource-private-ip-address": "'${RESOURCEIPADDRESS}'"
      },
  "waitForState": [
    "SUCCEEDED"
  ],
  "waitIntervalSeconds": 60
}'  >$TEMPFILE
```

```
#
# Create the session
#
#CSTATUS=`oci bastion session create --profile $OCIPROFILE --bastion-id --from-
json "file://${TEMPFILE}"`
CSTATUS=`oci bastion session create --profile $OCIPROFILE --from-json
"file://${TEMPFILE}"`
echo $CSTATUS | jq '.data."lifecycle-state"' |  grep \
'ACCEPTED\|CANCELED\|CANCELING\|IN_PROGRESS\|SUCCEEDED\|ACTIVE\|CREATING'
if [ $? -ne  0 ]
then
  echo $CSTATUS
  echo ""
  echo "Creation of session failed"
  exit 1
fi
#
# check if creation failed
#
ASTATUS=`echo $CSTATUS | jq '.data."lifecycle-state"' | sed 's/"//g'`
if [ "$ASTATUS" = "FAILED" ]
then
  echo $CSTATUS
  echo "Creation of session failed"
  exit 2
fi
#
# grab the ID of the session, and check if it is active
#
ID=`echo $CSTATUS | jq '.data.id' | sed 's/"//g'`
echo "ID: "$ID
if [  "$ASTATUS" != "ACTIVE" ]
then
    echo "waiting for session creation (approx 60 sec.)"
    for I in 1 2 3 4 5 6
    do
      sleep 10
      ASTATUS=`oci --profile $OCIPROFILE bastion session get --session-id $ID |
jq '.data."lifecycle-state"' | sed 's/"//g' `
      if [ "$ASTATUS" = "ACTIVE" ]
      then
        echo "bastion service is ready"
        break
      fi
      echo "Creation status: "${ASTATUS}
    done
     if [ "$ASTATUS" != "ACTIVE" ]
      then
        echo "bastion service is not yet available "
        echo "Please verify with command: "
        echo "oci --profile $OCIPROFILE bastion session get --session-id $ID"
        exit 1
      fi
fi #
#  Connect to the session
#
ssh -i ${SSHKEYDIR}${SSHPRIVATEFILE} -o ProxyCommand="ssh -i \
${SSHKEYDIR}${SSHPRIVATEFILE} -W %h:%p -p 22 \
${ID}@host.bastion.${OCIREGION}.oci.oraclecloud.com" \
 -p ${RESOURCEPORT} ${OSUSERNAME}@${RESOURCEIPADDRESS}
```

Connect with us

Call +**1.800.ORACLE1** or visit **oracle.com**. Outside North America, find your local office at: **oracle.com/contact**.

blogs.oracle.com facebook.com/oracle twitter.com/oracle