



Version
6.13

Financial Crime Risk Management Suite

INSTALLATION GUIDE

Redhat Openshift OCP

Revision A

Legal Notice:

This document is proprietary & confidential and may only be used by persons who have received it directly from ThetaRay LTD. ("ThetaRay") and may not be transferred to any other party without ThetaRay's express written permission. The document provides preliminary and general information only and is not intended to be comprehensive or to address all the possible issues, applications, exceptions or concerns relating to ThetaRay. All information contained in this document is confidential and shall remain at all times the sole property of ThetaRay. The recipient of this document has no right to disclose any or all of its contents or distribute, transmit, reproduce, publicize or otherwise disseminate this document or copies thereof without the prior written consent of ThetaRay, and shall keep all information contained herein strictly private and confidential. The information is intended to facilitate discussion and is not necessarily meaningful or complete without such supplemental discussion. Please note that the information procedures, practices, policies, and benefits described in this document may be modified or discontinued from time to time by ThetaRay without prior notice. As such, ThetaRay provides the document on an "As-Is" basis and makes no warranties as to the accuracy of the information contained therein. In addition, ThetaRay accepts no responsibility for any consequences whatsoever arising from the use of such information. ThetaRay shall not be required to provide any recipient with access to any additional information or to update this document or to correct any inaccuracies herein which may become apparent.

1. Introduction	7
2. Deployment Prerequisites	8
2.1. Namespaces	8
2.1.1. Cluster default LimitRanges (Openshift only)	8
2.2. Compute	9
2.2.1. On cloud	9
2.3. Storage	11
2.3.1. Minio Large Volumes Procedure(Openshift Only)	11
2.4. Networking	13
3. Deployment Preliminary Steps	15
3.1. Bastion Server Setup	15
3.2. Images	15
3.3. Configure Environment Specifics	16
3.4. Namespaces Setup	16

3.4.1. Set docker Registry and CA cert if Needed	17
3.4.2. applications (IC) package	18
3.4.3. SLA Display in Days	18
3.4.4. Secrets	18
3.4.5. Additional Deployment Configurations	19
3.5. Encrypting Password in Secrets	19
3.5.1. Requirements	19
3.5.2. Default Environment	19
3.5.3. Sampling Predefined Scripts	20
3.5.4. For Production Use or Production Flow Simulation	21
3.5.5. How to Re-encrypt Files with New Secret Key?	21
4. Deployment	22
4.1. Deployment of Individual Namespaces	22
4.1.1. Install the Shared Infrastructure (Infra)	23
4.1.2. Install the Platform	23
4.1.3. Install Investigation Center (IC)	23
4.2. Deploying Using the Offline Installer	23
4.2.1. Overview	23
4.2.2. Prerequisite	24
4.2.3. Accessing and Installing Config Files	24
4.2.4. Key Attributes and Description	24
4.2.5. Supplementary Python 3 Installation Instructions	26
4.2.5.1. Installing Python 3	26
4.2.5.2. Verification steps	26
4.3. IC - Global Trace Query Limit Configuration Procedure	27
4.3.1. Deployment Configuration - Overview	27
4.4. Highly Available Deployment - Microsoft Azure (Openshift Only)	27
4.4.1. Overview	27
4.4.2. Multi - AZ Support	28
4.4.3. Enabling High Availability Option	29
4.4.4. Setting the Number of Replicas for Components in the Application Namespace	29

4.4.5. ZRS Storage Class Setting	29
5. Post Deployment	31
5.1. Running Sanity & Clean up	31
5.1.1. Prerequisite:	31
5.1.2. Main Steps	32
5.2. Enabling Alerts Messaging Via Email	34
5.3. Licensing Procedure Instructions	36
5.3.1. Resources	36
5.3.2. Procedure Steps	36
6. Upgrade to 6.11	39
6.1. Upgrade Preliminary Steps	39
6.1.1. Bastion Server prerequisites	39
6.1.2. Images	39
6.1.3. Configure Environment Specifics	40
6.1.4. Run Upgrade on Openshift	41
7. Post-upgrade	45
7.1. Basic Assumptions for Health Check	45
7.2. What does the Health Check cover?	45
7.2.1. Running Sanity after upgrade	45
7.2.2. How to run Health Check	46
7.3. Licensing Procedure Instructions	49
7.3.1. Resources	49
7.3.2. Procedure Steps	49
8. Appendix A - Data Tiering	52
8.1. Introduction	52
8.2. Azure Prerequisites	52
8.3. Deployment Parameters	52
9. Appendix B - Data at Rest Encryption	54
9.1. Using Customer Provided Data Encryption Key (DEK)	55
10. Appendix A - Backup and Disaster Recovery	56
10.1. Introduction	56
10.2. Backup, Restore and Disaster Recovery	57

10.2.1. Purpose and Scope	57
10.2.2. Overview	57
10.2.3. Backup	58
10.2.4. Prerequisites for Setting up Backup Targets	58
10.2.4.1. Google Cloud Storage	58
10.2.4.2. Azure Blob Storage	59
10.2.5. S3 Compatible Storage	60
10.2.6. Configuration & Installation	61
10.2.6.1. Configuration	61
10.2.6.2. Installation	63
10.2.7. Backup Running Process Overview	64
10.2.8. Backup Components Overview	64
10.2.8.1. PostgreSQL	65
10.2.8.2. OpenSearch	66
10.2.8.3. GitLab	68
10.2.9. Backup to External Storage	70
10.2.10. Manual Backups	74
10.2.10.1. Internal Platform Backups	75
10.2.10.2. External Backups	78
10.2.11. Restore from Backup	83
10.2.12. Point in time restore of external storage	90
10.2.12.1. How can I know which point of time I want to restore? (logging)	90
10.2.12.2. How can I perform PITR?	92
10.2.13. Disaster Recovery	95
10.2.14. Setup Environment	95
10.2.15. Recovering Environment	97
10.2.16. Disaster Recover (DR) Summary	100
11. Appendix D - Prerequisites - OCP Specific	101
12. Appendix E - Specification	123
13. Appendix F - Reporting Database Exposure	126
13.1. Introduction	126

13.2. Configuring the Deployment to Expose the Reporting Database	126
14. Appendix G - Enable SLA in Days	128
14.1. Introduction	128
14.2. Enablement Process	128
15. Appendix H - Toggle between CRA 1 and CRA 2	129
15.1. Introduction	129
15.2. CRA 1 - 2 Toggling	129
16. Appendix J - Create TM Manual Alert when no Activity	130
16.1. Introduction	130
16.2. Instructions	130
17. Appendix K - Rule Builder & Simulator	131
17.1. Deployment	131

1. Introduction

This install guide describes the essential components and commands used in the installation of this version of the ThetaRay Platform and Investigation Center module.

The main stages of the installation are covered in the following chapters:

- Deployment Prerequisites
- Deployment Preliminary Steps
- Deployment
- Post Deployment
- Upgrade
- Sanity Health Check
- Data Tiering
- Back up, Restore and DR
- Prerequisites - OCP specific

2. Deployment Prerequisites

Note: Please consult the relevant version release notes for the specific OpenShift / Azure Kubernetes Service.

2.1. Namespaces

The deployment of the ThetaRay 6.x system spans multiple Kubernetes / OpenShift namespaces, inter-communicating with one another (in case Network Policies are used to restrict traffic within the cluster - an administrator should ensure that traffic can flow between Pods in the different namespaces)

The basic deployment spans 4 namespaces that need to be created by a cluster administrator as follows:

- **<Infrastructure>** - infrastructure components shared by all other components including data stores, version control system & monitoring infrastructure.
- **<Platform>** - a namespace hosting platform components dealing with automated / interactive data processing and analysis. This includes components such as Airflow & Jupyter. This namespace includes statically deployed services (e.g. Airflow Web Server, Jupyter Hub) & dynamically launched Pods (Jupyter notebook servers & Airflow Executor Pods).
- **<Executors>** - a namespace used for hosting dynamically launched Spark Executor Pods.
- **<Applications>** - a namespace hosting Investigation Center frontend and backend micro services.

Multiple Platform Instances can be attached to the infrastructure namespace, to support data / projects segregation. In the case of multi-instance deployment, each Platform Instance will be associated with a pair of 'Platform' / 'Executors' namespaces.

The system supports distributing alerts from multiple Investigation Center instances, this is facilitated by deploying multiple instances of the 'Investigation Center' across multiple 'Applications' namespaces, all attached to the same 'Infrastructure' namespace.

2.1.1. Cluster default LimitRanges (Openshift only)

Recommended Values

```
spec:
  limits:
    - type: Container
      max:
        cpu: '8'
```



```

    memory: 32Gi
  min:
    cpu: 10m
    memory: 4Mi
  default:
    cpu: '4'
    memory: 4Gi
  defaultRequest:
    cpu: '1'
    memory: 1Gi
  maxLimitRequestRatio:
    cpu: '99'

```

2.2. Compute

ThetaRay system is deployed into a Kubernetes / Openshift environment and supports dynamic scaling out / in, based on compute resources needs. The default system sizing assumes worker nodes each having 16 VCPUs and 64GB of memory with the maximum number of nodes required determined by transaction volumes and data processing complexity.

The systems spins up / tears down Kubernetes Pods based on dynamic compute requirements ranging from establishing Jupyter based notebook environments for data science interactive work, to dynamically launching Spark Executors for handling data processing activities.

2.2.1. On cloud

On cloud (either public or private) environments that support cluster auto scaling (i.e. dynamically provisioning virtual machines based on Pod resources requirements), this dynamic behavior allows the system to dynamically acquire cloud resources when activities are running and release them once completed.

The system supports ensuring that the ThetaRay deployment will run on specific nodes within the cluster through a configurable 'Node Selector', allowing an administrator to configure the container platform to host Pods on nodes with specific labels.

To support the above autoscaling behavior on Red Hat OpenShift, the cluster should be pre-configured to support autoscaling (see - https://docs.openshift.com/container-platform/latest/machine_management/applying-autoscaling.html)

1. Configure cluster level 'ClusterAutoScaler' for enabling dynamic cluster auto scaling (see - https://docs.openshift.com/container-platform/latest/machine_management/applying-autoscaling.html#configuring-clusterautoscaler)
2. Configure 'MachineSet' which will be used to host ThetaRay workloads (if needed multiple MachineSets can be set up to handle different platform instances) (see - https://docs.openshift.com/container-platform/latest/machine_management/creating_machinesets/creating-machineset-azure.html)

3. Configure a 'MachineAutoscaler' to configure auto scaling behavior for the MachineSet defined in (2) (see - https://docs.openshift.com/container-platform/latest/machine_management/applying-autoscaling.html#configuring-machineautoscaler)

When deploying into an Azure Kubernetes Service the system assumes that two Node Pools are available and are configured to auto scale nodes. The system workloads are categorized into two types –

- Services -> Pods that are continuously running. These will be running on a Node Pool with a virtual machines having the label 'tr-type: applications'. In a typical deployment nodes within the services node pool should consist of 8 CPU cores and 32 GB of memory.
- Executors -> Pods that are spinned up on demand, such as Spark or Airflow executors. Nodes spun up by this Node Pool should be assigned with the label 'tr-type: executors'. In a typical deployment nodes within the executors node pool should consist of 16 CPU cores and 64GB of memory.

2.3. Storage

ThetaRay system relies on Read Write Once Kubernetes Persistent Volumes dynamically established during Stateful Sets / Pods creation.

Persistent Volumes may be created :

- At deployment time for individual Pods within a Stateful Sets (e.g. Postgres, Minio). These volumes are persistent and are associated with their respective Pods even in case of eviction and re-establishment of Pods on alternate nodes.
- For Redhat Openshift - Temporary volumes established during creation of Spark Executor Pods - the lifetime of the volumes is associated with the lifetime of the associated Pod. When running on Azure Kubernetes Service, the system makes use of local disks for temporary data storage (when running on a general purpose DS16v3 VM with 16GB, this implies a 128GB) by default instead of utilizing Persistent Volumes.

Volume management is handled by the application either during the deployment process or during runtime with no need for user intervention.

To accommodate additional storage needs without having to scale out the cluster or going through backup / restore exercises, the Kubernetes / OpenShift Storage Class used for ThetaRay volumes should be configured to support volume expansion. This is facilitated by configuring using a Storage Class with volume expansion enabled.

For Redhat Openshift, see - <https://docs.openshift.com/container-platform/latest/storage/expanding-persistent-volumes.html> for additional details.

2.3.1. Minio Large Volumes Procedure(Openshift Only)

By default, Openshift is performing a recursive persistent volume scan on Pods startup in order to change file ownership and SELinux file labels. In case of MinIO Pods this may result in slow Pod startup or even timeouts when the object storage consists of a large number of files (the order of magnitude of hundreds of thousands).

To mitigate this by instructing Openshift to skip this process, Minio should run under a specific Security Context Constraint detailed below -

Important: In the case where "Minio large files" is used AND SELinux is **disabled**, make sure to remove the value "spc_t" from the installation files, example:

```
seLinuxOptions:
  type: spc_t <---Remove
```

Path to file to be modified:

```
helm/data-access-service/templates/statefulset.yaml
```

Large Volumes with Minio

The Security Context Constraint should be registered by a cluster administrator prior to deployment / upgrade of the ThetaRay system

```

1  allowHostDirVolumePlugin: false
2  allowHostIPC: false
3  allowHostNetwork: false
4  allowHostPID: false
5  allowHostPorts: false
6  allowPrivilegeEscalation: true
7  allowPrivilegedContainer: false
8  allowedCapabilities: NET_BIND_SERVICE
9  apiVersion: security.openshift.io/v1
10 defaultAddCapabilities: null
11 fsGroup:
12   type: MustRunAs
13 groups:
14 - system:authenticated
15 kind: SecurityContextConstraints
16 metadata:
17   name: trminiolargevolume
18 priority: null
19 readOnlyRootFilesystem: false
20 requiredDropCapabilities:
21 - KILL
22 - MKNOD
23 - SETUID
24 - SETGID
25 runAsUser:
26   type: MustRunAsRange
27 seLinuxContext:
28   type: RunAsAny
29 supplementalGroups:
30   type: RunAsAny
31 users: []
32 volumes:
33 - configMap
34 - downwardAPI
35 - emptyDir
36 - persistentVolumeClaim
37 - projected
38 - secret

```

The Security Context Constraint should be registered by a cluster administrator prior to deployment / upgrade of the ThetaRay system.

To enable usage of the custom SCC by the service account running Minio' the common.yaml file under the ocp-prod environment should be configured with the following settings -

```
minio_large_volume:
  enabled: true
  scc_name: trminiolargevolume
```

The 'scc_name' provided avoids configuration and should be aligned with the name assigned by the cluster administrator.

For more details around the operation please refer to the Redhat Openshift support site -

<https://access.redhat.com/solutions/6221251>

2.4. Networking

While Pods within a Kubernetes cluster can easily communicate between themselves, they are not by default accessible to external networks and traffic. A Kubernetes Ingress is an API object that shows how traffic from the internet should reach internal Kubernetes cluster Services that send requests to groups of Pods. The Ingress itself has no power. It is a configuration request for the ingress controller that allows the user to define how external clients are routed to a cluster's internal Services. The ingress controller hears this request and adjusts its configuration to conform with the user's requirement.

Note: Cluster Ingress is not managed by ThetaRay.

Ingress Traffic

ThetaRay system exposes multiple HTTPS based endpoints through the cluster's Ingress Controller. These endpoints expose both web based user interfaces for the various system components as well as endpoints exposing REST / GraphQL based APIs enabling data / process level integrations. Each of these endpoints is associated with a unique FQDN configurable as part of the system setup.

To enable system access the IP address (or addresses) associated with the cluster's Ingress Controller should be registered for each of these FQDNs in the corporate DNS. DNS resolution for these FQDNs should be made available both for external clients and for Pods inside the cluster.

Certificates for the relevant FQDN should be provided either as a wildcard certificate or as an individual per host certificate.

Following is the list of exposed ingresses:

- Jupyterhub (default - jupyterhub-<platform namespace>.<cluster domain>) - interactive data science notebooks
- MLFlow (default - mflow-<platform namespace>.<cluster domain>) - model management and experiment tracking interface
- Airflow (default - airflow-<platform namespace>.<cluster domain>) - Job automation
- Spark History Server (default - spark-<platform name>.<cluster domain>) - Batch jobs execution tracking
- Gitlab - (default - gitlab-<infra namespace>.<cluster domain>) - Git web interface / HTTPS interface to Git
- Hasura - (default - graphql-<infra namespace>.<cluster domain>) - GraphQL data gateway
- Keycloak - (default - keycloak-<infra namespace>.<cluster domain>) - Identity management & access control
- Minio - (default - minio-<infra namespace>.<cluster domain>) - S3 compatible object storage API
- Minio Console - (default - minio-console-<infra namespace>.<cluster domain>) - Web interface for Minio
- Monitoring - (default - monitoring-<infra namespace>.<cluster domain>) - Web interface for alert management & event exploration
- Pgadmin4 - (default - pgadmin4-<infra namespace>.<cluster domain>) - Postgres administration UI
- Investigation Center
- API Gateway

Traffic to Ingress from Pods

Multiple Pods with the ThetaRay environment communicate directly with the cluster Ingress Controller - firewall rules should be set up to enable this traffic.

Egress Traffic (optional)

Note: It is important to point out, that the 'egress' configuration described here, is *optional* and not a required *mandatory* configuration.

The system can optionally be configured to forward detected alerts to a remote HTTPS endpoint(s) accepting data in a JSON format mandated by ThetaRay . If used, the cluster environment should be configured to enable DNS resolution of the remote endpoint as well as network connectivity to the relevant host.

The full list of OCP prerequisites is listed in [Appendix D](#)

3. Deployment Preliminary Steps

3.1. Bastion Server Setup

The bastion server that will be used for installing ThetaRay requires the following packages:

- helm 3 ($\geq 3.8.2$)
- age
- helmfile
- helm-diff
- helm-secrets
- kubectl
- oc (openshift client)
- mc (minioclient)
- jq
- sops
- git-lfs (necessary for running export with model)

For secret encryption, you should install the packages as described in [Encrypting Password in Secrets](#)

3.2. Images

Download the ThetaRay installation package to the Bastion server and extract the images tar file:

```
tar xzf tr-platform-images.tgz
tar xzf tr-applications-images.tgz
tar xzf tr-static-images.tgz
```

Load all the images locally with:

```
docker load -i <filename.tar>
```

Tag all the images before pushing to the docker registry:

```
docker tag thetaray/image:tag <docker registry>/thetaray/image:tag
```

If required, login to the registry with a username and password:

```
docker login <docker registry>
```

For more information:

<https://docs.docker.com/engine/reference/commandline/login/>

Push the images to the docker registry:

```
docker push <docker registry>/thetaray/image:tag
```

3.3. Configure Environment Specifics

Download and extract ThetaRay helm chart tar files:

(replace the x with the actual build number)

```
tar xzf tr-platform-6.*.tgz
tar xzf tr-applications-6.*.tgz
```

3.4. Namespaces Setup

Login to Kubernetes environment. For OpenShift cluster:

```
oc login --server=https://<OpenShift server>:6443
```

For Azure AKS copy and paste the login commands from the Azure Portal.

Create the 4 namespaces namespace:

```
oc new-project <infra namespace name> #
use kubectl in case of AKS
oc new-project <platform namespace name>
# use kubectl in case of AKS
oc new-project <exec namespace name> #
use kubectl in case of AKS
oc new-project <applications namespace
name>
# use kubectl in case of AKS
```

In the case where a secured docker-registry is used for Openshift, create the secret containing the credentials in all 4 namespaces:

For example:


```
oc -n <namespace name> create secret docker-registry <secret-name> --docker-
server=https://<your-registry-server> --docker-email=user@example.com --
docker-username=<your-username> --docker-password=<your-password>

oc -n <namespace name> secrets link default <secret-name> --for=pull
```

In case of using an internal root CA or a self signed certificate, create a secret that contains the CA bundle in all 4 namespaces:

The key inside the secret must be equal to "ca-bundle.pem" (don't change the "--from-file=ca-bundle.pem")

```
oc -n <namespace name> create secret generic ca-bundle-secret --from-file=ca-
bundle.pem=/path/to/file/ca-bundle.crt
```

extra-values

platform package:

Choose your env profile (e.g. ocp-production) and edit the environment files.

Set cpu + memory requests and limits, volumes sizes and node selector for application and executors inside the following files:

- environments/ocp-prod/shared-resources.yaml.gotmpl
- environments/ocp-prod/solution-resources.yaml.gotmpl

Note: At the file solution-resources.yaml.gotmpl you will find that 2 node selectors are provided to provide the ability to split resources between application and executors. In case you do not need it, just put same labels there.

3.4.1. Set docker Registry and CA cert if Needed

environments/ocp-prod/common.yaml.gotmpl

values:

image_registry: <registry host-name>

image_registry_pull_secret: <a secret to authenticate with the docker registry>

ca_bundle_secret: <in case of using a self signed certificate or an internal root CA>

hasura_ui: true/false <True is the default used for OCP offline installation>

In case of using a specific storage_class, update:

environments/ocp-prod/solution-resources.yaml.gotmpl

value:

```
spark_executor:
  storage_class: <storage class>
```

3.4.2. applications (IC) package

```
helm/values/common-values.yaml
```

values:

```
dockerRegistry: <registry host-name>
dockerRegistryPullSecret: ""
platform: "ocp"
clusterDomain: "<cluster domain>"
icInstanceName: "<apps_namespace>"
sharedNamespaceName: "<infra-namespace>"
s3BucketName: "<apps-namespace>"
envprofile: "prod"
nodeSelector: {}
screening:
  enabled: true/false
```

3.4.3. SLA Display in Days

For details on how to change the default SLA (hours to days) refer to Appendix G.

3.4.4. Secrets

Set the passwords of your choice in:

platform package:

```
environments/ocp-prod/shared-secrets.yaml.gotmpl
```

Note: Value of key "opensearch_admin_password" should not contain any special symbols. Use only letters.

```
environments/ocp-prod/solution-secrets.yaml.gotmpl
```

applications (IC) package:

```
helm/values/secrets.yaml
```

3.4.5. Additional Deployment Configurations

In the deployment files, it is possible to modify the number of days that will be used as versions retention policies in MinIO (ilm).

This is the number of days after which versions of an object will be deleted.

» To modify this setting:

1. Navigate to, and open the *common.yaml.gotmpl* file in your profile.
2. Locate the variable "*minio_ilm_versions_retain_days*". By default it's set at 2 days which is the recommended value as shown in the following code snippet

```
minio_ilm_versions_retain_days: 2
```

3. If required to modify this, change the value shown.
4. Make sure you **save** before closing the file.

3.5. Encrypting Password in Secrets

3.5.1. Requirements

- age: simple encryption tool. Used only for creating key pair file:
<https://github.com/FiloSottile/age>
- sops: editor of encrypted files that support different encrypts backend (AWS KMS, GCP KMS, Azure Key Vault, age, PGP):
<https://github.com/mozilla/sops>
- helm-secrets helm plugin that can securely decrypt encrypted file on the fly during chart deploy:
<https://github.com/jkroepke/helm-secrets>

3.5.2. Default Environment

The files in ocp-production and rke-production profiles are already encrypted and AGE-SECRET-KEY hardcoded inside each install script.

For the default installation flow, there no additional tasks to be undertaken.

How to *encrypt/decrypt* or change default values in encrypted files?

Before you begin to work with **sops** and **age** the concept of public and secret keys requires to be clarified..

Public key: (also known as 'recipient') is the open key that is used for encrypting files. this can be shared freely. This key is always exported to ***SOPS_AGE_RECIPIENTS*** variable before encryption.

Secret key: is used for decryption and should be kept in secret. This key is always exported to ***SOPS_AGE_KEY*** variable before decryption or deployment.

If a unique key is not used during installation, make sure that key is exported. To do this, run the script under the sops folder as follows:

1. Run script sops_cmd.sh
2. Select option 7.
3. Type `"/age-key.txt"` (or the location of the file).
4. Click **Enter**.

3.5.3. Sampling Predefined Scripts

To ease working with ***encrypt/decrypt/edit*** and other operations, you can use the scripts stored inside the ***detection-platform/sops*** folder.

These scripts already have all default dev env variables exported inside them, allowing you to perform operations interactively.

sops_cmd.sh - probably the most commonly used commands that you may need when working with ***age*** and ***sops***.

Note: All the following listed commands are executed against one file at a time.

Option	Command Name	Description
1	decrypt file	Used to decrypt file
2	encrypt file	Used to encrypt file
3	edit encrypted file	Used to edit an already encrypted file
4	generate age (plain) keyfile	Plain text
5	generate age keyfile	encrypted with password
6	decrypt age keyfile to stdout	decrypted the keyfile to stdout
7	generate sops vars	from plain age keyfile
8	generate sops vars	from encrypted age keyfile
9	unset sops vars	Unset variables
10	quit	

decrypt_all.sh - decrypt all encrypted files in ocp-production and rke-production profiles (by all files, means those files that are specified in the helmfile "secrets:" block)

encrypt_all.sh - encrypt all plain files in ocp-production and rke-production profiles (by all files, means those files that are specified in the helmfile "secrets:" block)

create_encrypted_age_keyfile.sh - create encrypted password protected age key pair file.

3.5.4. For Production Use or Production Flow Simulation

Following are examples of working from the console.

Encrypt file:

```
export SOPS_AGE_
RECIPIENTS='age1ccuz5mfc9nsukjlefqnnvkt2yk3q97a7fewut749ujhpt3zlegtqxvakuq'
sops -e -i plain_file_name.yaml
```

Decrypt file:

```
export SOPS_AGE_KEY='AGE-SECRET-KEY-
1D76RP8UT5HNQRXHKDY2D85SMRV8WNJJHDJXQ9S46T6SXHR2XVS0QD456DQ'
sops -d -i encrypted_file_name.yaml
```

Edit encrypted file: (will open default console editor, after save all changes will be encrypted automatically):

```
export SOPS_AGE_KEY='AGE-SECRET-KEY-
1D76RP8UT5HNQRXHKDY2D85SMRV8WNJJHDJXQ9S46T6SXHR2XVS0QD456DQ'
sops encrypted_file_name.yaml
```

3.5.5. How to Re-encrypt Files with New Secret Key?

Once again, this can be done by using the script inside detection-platform/sops folder.

» The scripts order is as follows:

1. Decrypt all files with decrypt_all.sh
2. Create new age key pair file. Use sops_cmd.sh (menu 4 or 5)
3. Generate env variables. Use sops_cmd.sh (menu 7 or 8 based on what was chosen in previous step) Execute generated commands in console.
4. Encrypt all files with encrypt_all.sh
5. Before deploying, make sure that the SOPS_AGE_KEY variable (from 3 step) exists in the current console window.

4. Deployment

Two flavors of installations of scripts can be used to deploy a ThetaRay environment.

- Low level shell scripts / direct use of the 'helmfile' tool to deploy individual namespaces (shared, platform and applications) and manual invocation of sanity test jobs. These require manual maintenance of command line arguments and merging of parameter files between releases.
- A higher level 'offline installer' that orchestrates the deployment across multiple namespaces, performs automated parameters merge operations and maintains the current environment configuration as a Git project.

A ThetaRay environment can be deployed into various types of infrastructure. The type of infrastructure into which the environment is deployed is identified by the env-profile parameter / configuration file settings with the supported values being:

- ocp-production – Redhat Openshift running either on-prem (virtualized / bare metal) or on Microsoft Azure
- private-cloud – deployment into a customer provided Azure Kubernetes Service Environment
- rke-prod – deployment into a Rancher RKE Kubernetes environment deployed by ThetaRay and running on customer premises

The following sections provide details on how to invoke the two flavors of the deployment.

4.1. Deployment of Individual Namespaces

Note: The Platform should be installed before the app, and they should not be run in parallel.

The following section includes instructions on how to invoke the shell scripts / tools from the bastion server when deploying into each namespace.

4.1.1. Install the Shared Infrastructure (Infra)

run the installation script

```
./install-shared.sh --new-shared-namespace <infra namespace name> --env-profile <env profile> --cluster-domain <cluster domain> [--create-namespace false] --customer <customer-bucket-name>
```

Note: <customer-bucket-name> - is a customer identifier, that should be set on installation.

It is used as a part of the backup bucket, to make it unique.

Verify all Pods are running and the script finished successfully.

4.1.2. Install the Platform

Run the installation script:

```
./install-env.sh --solution <solution name> --namespace <platform namespace name> --shared-namespace <infra namespace name> --exec-namespace <exec namespace name> --env-profile ocp-production --cluster-domain <cluster domain> [--create-namespace false]
```

After running the above commands wait for the pods to restart, and verify all Pods are running and the script finished successfully.

4.1.3. Install Investigation Center (IC)

Run the installation:

```
[CREATE_NAMESPACE=false] helmfile --namespace <apps namespace name> -e ocp-production -f helmfile-applications.yaml apply
```

Verify all Pods are running and the script finished successfully.

4.2. Deploying Using the Offline Installer

4.2.1. Overview

The Offline Installer is a utility that enables automated orchestration of a full system deployment including execution of post installation sanity jobs.

4.2.2. Prerequisite

Python -3 is a requirement for the offline installer. In case it is not installed as part of Linux deployment a manual installation procedure is required. Full installation instructions are provided at the end of this section.

4.2.3. Accessing and Installing Config Files

The files used to run offline installation or upgrade of the ThetaRay application in an on-premise customer environment are stored in the following path:

```
tr-platform package folder>/tools/offline-installer
```

4.2.4. Key Attributes and Description

installation-script.sh - main script

config_files/main.config - file that holds all configuration values for deploying system

auxiliary_scripts/*.sh - auxiliary scripts that are called from inside the main script. Each one is designed to do only one function..

Note: These scripts can't be executed as standalone scripts.

auxiliary_scripts/supplemental_files - non script files (txt,yaml,json etc) that are needed to enable the installation.

script_logfiles/*.log - Log file.

Installation Script

installation-script.sh or main script is interactive and designed to be executed without need to pass shell arguments from user side. All information being taken from text based config file: ***config_files/main.config***

Config Files

config_files/main.config files are constructed from key value pairs where each key/value pair represent one setting at a time, and has description and example.

Important: By default the config file is shipped with empty values.

Example of populated file with values (without comments):

```
CFG_PLATFORM="ocp"

CFG_SHARED_NS_NAME="infra-prod"
CFG_PLATFORM_NS_NAME="platform-prod"
CFG_EXEC_NS_NAME="exec-prod"
CFG_APPS_NS_NAME="app-prod"

CFG_SUFFIX="prod"
CFG_CUSTOMER="trust-bank"
CFG_SOLUTION="production"
CFG_CLUSTER_DOMAIN="trust-bank.com"
CFG_MULTIPLE_REALM="false"
CFG_HIGH_AVAILABILITY="false"
CFG_IMAGE_REGISTRY="trust-bank.io/thetaray"
CFG_IMAGE_REGISTRY_PULL_SECRET="registry-creds"
CFG_CA_BUNDLE_SECRET="internal-certs"
CFG_GLOBAL_NODE_SELECTOR='
node-type: "internal"
env: prod
disk-type: "ssd-generic"'

CFG_NODE_SELECTOR_EXECUTORS='
node-type: "on-demand"
env: prod'

CFG_OLD_PACKAGE_SOURCE="filesystem"
CFG_OLD_PLATFORM_PACKAGE_LOCATION="/opt/tr-platform-production.18"
CFG_OLD_APPLICATION_PACKAGE_LOCATION="/tmp/old/tr-application-production.15"
```

Each run of the script creates logfile inside **./script_logfiles** folder with unique templated name: *"\$Year-\$Month-\$Day_\$Hour_\$Minute_\$Second.log"*

Note: In case of errors, this log can be shared with Thetaray Support.

Message Types

During execution of main script, messages on a screen will be shown in different colours to highlight different action.

- **[INFO] Message text** - **Yellow** text print information messages during execution to show what action is running at this moment, what will be executed next or any kind of intermediate results
- **[WARN] Message text** - **Light red** messages attract user attention to some result during execution that are important and that can have impact on deployment process
- **[ERROR] Message text** - **Red** messages happens during non zero exit status of any action and always mean that fatal error occurred and further execution impossible
 - Message text - **Green** messages show that current action was finished successfully
 - Message text - **Purple** messages indicate a question that need to be answered by user to proceed
 - Message text - Default terminal colour messages that present other information (answer options, value settings etc)

4.2.5. Supplementary Python 3 Installation Instructions

4.2.5.1. Installing Python 3

The default Python implementation is usually installed by default. To install it manually, use the following procedure.

» To install Python 3.9, use:

```
# dnf install python3
```

» To install Python 3.11, use:

```
# dnf install python3.11
```

4.2.5.2. Verification steps

To verify the Python version installed on your system, use the `--version` option with the python command specific for your required version of Python.

For Python 3.9:

```
$ python3 --version
```

For Python 3.11:

```
$ python3.11 --version
```

4.3. IC - Global Trace Query Limit Configuration Procedure

For support personnel or on- prem deployment users who wish to adjust the deployment global trace query transaction query limit value.

4.3.1. Deployment Configuration - Overview

The json file that defines the trace query limitation is located in the frontend repository.

```
path: tr-applications-production/values/common-values.yaml
```

In the json, there is the row showing initially the default value:

```
"traceQueryCountLimit"=50000
```

1. If needed to change, change this count before deployment
2. In order to change limit after application is deployed, or as part of a version upgrade, the engineer needs to get inside frontEnd pod and edit the configmap config.json

Example:

- a. `kubectl edit configmap config.json -n <namespace of application>`
- b. Then change the value as in pre - deployment as detailed above.

4.4. Highly Available Deployment - Microsoft Azure (Openshift Only)

4.4.1. Overview

The ThetaRay system supports high availability option. The deployment topology assumes that the Openshift deployment spans 3 Availability Zones located within a single Azure Region, in order to ensure system availability in the case where a

whole Azure Availability Zone is not available.

When deployed in 'high availability mode', the system relies on data replication cross zones, and uses application level replication for major data stores (Postgres, Minio, Opensearch) combined, with storage level redundancy (Zone Redundant Storage or ZRS) for stateful components that do not have built in replication support (this is used only for components with minimal state).

Moreover, components which can benefit from multiple concurrent replicas are deployed with multiple concurrent instances, allowing both fast failure and when applicable, the ability to scale out.

Note: Multi AZ is only supported on OCP 4.10 (or higher), installations, on Azure public cloud with regions where ZRS disks exist.

4.4.2. Multi - AZ Support

Prerequisites for highly available, multi availability zone deployment

1. An Openshift version 4.10 (or higher), environment deployed across three Azure Availability zones, in an Azure region that supports Zone Redundant Storage (for example : West Europe).
2. Three Machinesets, each configured to manage nodes on a specific availability zone with a minimum of three nodes available per Machineset at deployment time. All Machinesets should share the same labels that will be used as a NodeSelector during ThetaRay deployment.
3. In order to use ZRS disks, StorageClass with ZRS option must be created before starting the installation.

Example for StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: zrs-standard-ssd
  annotations:
    storageclass.kubernetes.io/is-default-class: "false"
provisioner: disk.csi.azure.com
volumeBindingMode: WaitForFirstConsumer
allowVolumeExpansion: true
parameters:
  skuname: StandardSSD_ZRS
```

Once the Storage class is created, It should be used in the install process by changing the Storage Class: default / standard to the one that was created. Storage class option can be found under the relevant install profile config files for both parts of the system (Platform / Application)

4.4.3. Enabling High Availability Option

In order to deploy application in HA we added new flag to Install-shared.sh script named - ***high-availability true/false***

False - Install the Thetaray solution with default option.

True - Install the Thetaray solution with high availability option

4.4.4. Setting the Number of Replicas for Components in the Application Namespace

To work with High-availability option on APPs, please change the replicas for the following parts:

Resource Name	Replica Count
CRA	1
TR-Icfe	3
TR-Icbe	3
LogStash	3

Note: All config related to Apps can be found: `helm/values/common-values.yaml`

If high availability is set to true in the yaml file, then update the HPA yaml configuration as follows:

```

1 | icbe:
2 |   hpa:
3 |     minReplicas: 1
4 |     maxReplicas: 2

```

4.4.5. ZRS Storage Class Setting

Change Storage Class to ZRS for all workloads that use storage disk's. Using default, non zrs class will result in error similar to the following message:

```

"Disk /subscriptions/f711378a-2a82-4cf8-9c67-afac77fe459d/resourceGroups/OCP2-1X0Q2-RG/providers/Microsoft.Compute/disks/ocp2-1x0q2-dynamic-pvc-0af54542-b602-468b-b62b-f8ad6a34973f cannot be attached to the VM because it is not in the same zone as the VM. VM zone: '1'. Disk zone: '3'."

```

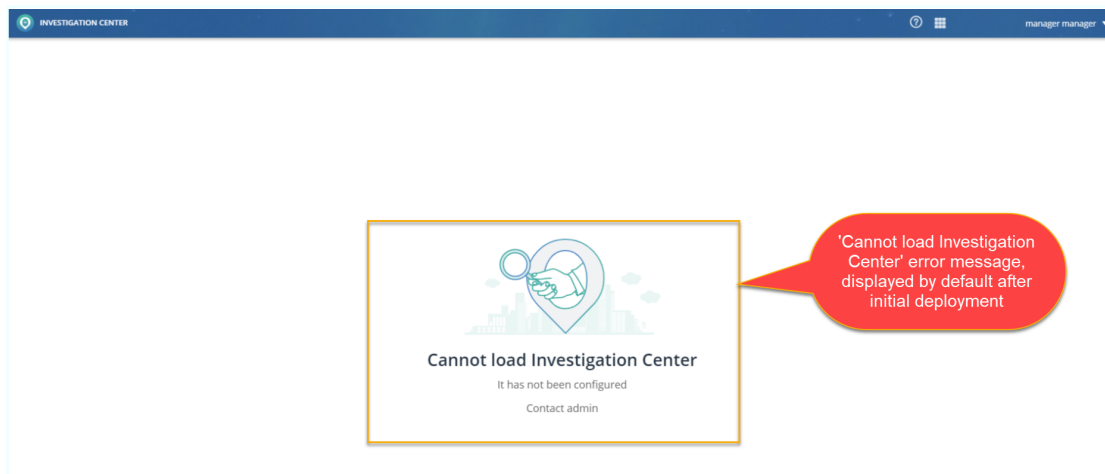
Further Storage Class Information

Change Storage Class to ZRS (Example name: zrs-standard-ssd) for the next components:

Resource Name	Value Under File
Redis	DP/environments/ocp-prod/shared-resources.yaml.gotmpl
Hub	DP/environments/ocp-prod/solution-resources.yaml.gotmpl
tr-platform	DP/environments/ocp-prod/solution-resources.yaml.gotmpl
rest-connector	DP/environments/ocp-prod/solution-resources.yaml.gotmpl
tr-CRA	Application/helm/environments/ocp-prod/resources.yaml
LogStash (All 3 of them)	DP/environments/ocp-prod/shared-resources.yaml.gotmpl Application/helm/environments/ocp-prod/resources.yaml
pgadmin4	DP/environments/ocp-prod/shared-resources.yaml.gotmpl

5. Post Deployment

Note: After Install is complete the IC (Investigation Center) will appear as not functioning when attempting to access the module, with the following popup being displayed:



This is due to a post deployment requirement to configure the IC Settings first.

5.1. Running Sanity & Clean up

Note the following:

- All tables created during sanity upgrade test execution are removed from postgres during the cleanup stage
- All records that are labeled as *%upgrade%* pattern will be deleted by the sanity clean script after running

Note: Running sanity should be for new deployments only. For systems that have been upgraded, execute **Health Check**.

5.1.1. Prerequisite:

- Verify 'sanityuser' is enabled in Keycloak, as indicated in the following figure:

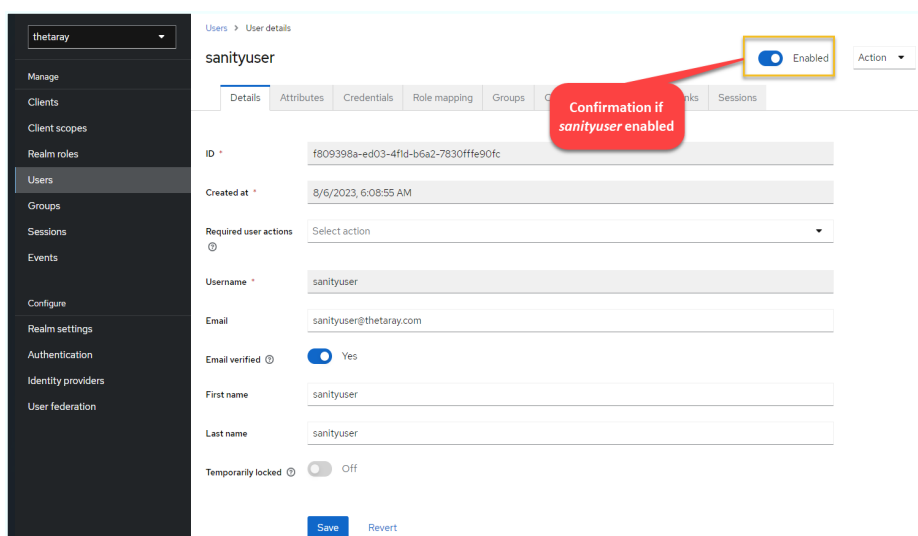


Figure 1: Sanity User enabled in Keycloak Verification

- Installation of the platform and investigation center is complete

The sanity deployment covers the following system parts

- Jupyter functionality and integration with GitLab, MLFlow and Postgres
- AirFlow functionality and integration with GitLab MLFlow and Postgres
- IC functionality and integration with the system
- The sanity run creates metadata and data under a special data permission, "dpv:sanity"

From version 6.1.1, note that all items related to sanity (e.g. datasets, evaluation flow and risks) are suffixed with *_upgrade*.

Note: *_upgrade* is a reserved term and therefore must **not** be used for naming entities.

5.1.2. Main Steps

The main steps in the test flow are:

1. Copying a new Sanity branch from Staging branch and backing up 'staging' and operational' branches.
2. Creation of metadata in the 'Sanity' branch
3. Execute basic Thetaray API using Jupyter
4. Merge 'sanity' to 'staging'
5. Execute basic Thetaray API using AirFlow
6. Execute basic sanity test on Investigation

The main steps in the cleanup flow are:

7. Delete 'sanity' git branch and restore 'staging' and 'operational'
8. Delete data in minio related to *_upgrade pattern*.
9. Delete data in spark related to *_upgrade pattern*.
10. Delete data in Postgres related to *_upgrade pattern*.
11. Delete alerts and settings in IC.

The main steps to run deployment sanity and clean up after Installation of the platform and investigation center are:

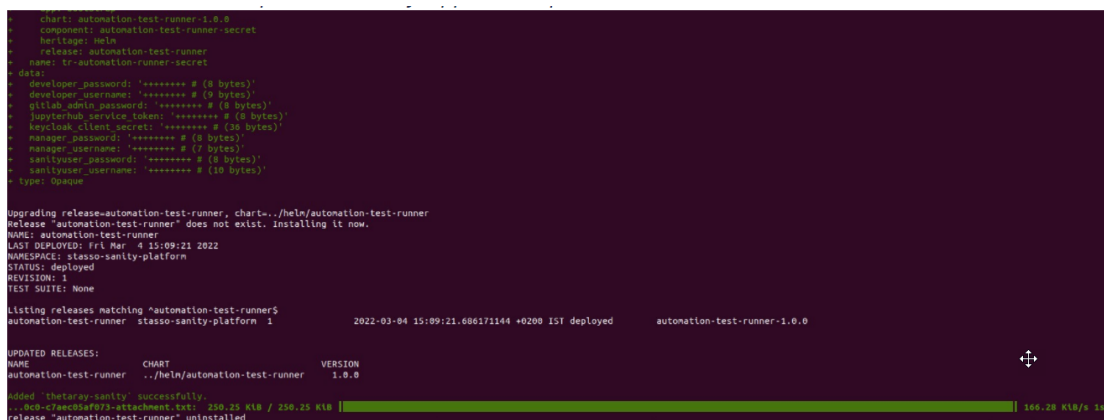
1. Go to the same directory where the platform installation script is located and run the sanity script with the RUN following option:

```
./sanity-runner.sh --solution <solution> --namespace <platform namespace> --
exec-namespace <exec namespace> --shared-namespace <shared namespace> --ic-
namespace <ic namespace> --ic-client-secret <ic-client-secret> --ic-instance-
name <ic_instance_name> --env-profile ocp-production --cluster-domain
<cluster domain> --create-namespace false --test-option RUN
```

Note: Take care when providing IC instance name, use the underscore character (_) and not the dash (-) character.

Example:

```
./sanity-runner.sh --solution test-sanity --namespace test-sanity-platform --
exec-namespace test-sanity-exec --shared-namespace test-shared-master--ic-
namespace test-sanity-app --ic-client-secret test-client-secret --ic-
instance-name test_instance_name --env-profile ocp-production --cluster-
domain test.sanity --create-namespace false --test-option RUN
```



```

> chart: automation-test-runner-1.0.0
> component: automation-test-runner-secret
> heritage: Helm
> release: automation-test-runner
> name: tr-automation-runner-secret
> data:
>   developer_password: '***** # (8 bytes)'
>   developer_username: '***** # (9 bytes)'
>   gitlab_admin_password: '***** # (8 bytes)'
>   jupyterhub_service_token: '***** # (8 bytes)'
>   keycloak_client_secret: '***** # (30 bytes)'
>   manager_password: '***** # (10 bytes)'
>   manager_username: '***** # (7 bytes)'
>   sanityuser_password: '***** # (8 bytes)'
>   sanityuser_username: '***** # (10 bytes)'
> type: Secret

Upgrading release automation-test-runner, chart ./helm/automation-test-runner
Release "automation-test-runner" does not exist. Installing it now.
NAME: automation-test-runner
LAST DEPLOYED: Fri Mar 4 15:09:21 2022
NAMESPACE: stasso-sanity-platform
STATUS: deployed
REVISIONS: 1
TEST SUITE: None

Listing releases matching "automation-test-runner$
automation-test-runner stasso-sanity-platform 1      2022-03-04 15:09:21.686171144 +0200 IST deployed automation-test-runner-1.0.0

UPDATED RELEASES:
NAME          CHART              VERSION
automation-test-runner  ./helm/automation-test-runner  1.0.0
Added 'thetarray-sanity' successfully.
... ReleaseManifests/2-attachment.txt: 250.25 KiB / 250.25 KiB [ ] 166.28 KiB/s 5s
release "automation-test-runner" uninstalled

```

Figure 2: Example verification of Sanity Run

2. To verify the test progress:

```
oc logs <automation-test-runner pod> -f -n <platform namespace>
```

3. Go to the same directory where the platform installation script is located and run the sanity script with the CLEAN following option:

```
./sanity-runner.sh --solution <solution> --namespace <platform namespace> --
exec-namespace <exec namespace> --shared-namespace <shared namespace> --ic-
namespace <ic namespace> --ic-client-secret <ic-client-secret> --env- profile
ocp-production --cluster-domain <cluster domain> --test-option CLEAN
```

Example:

```
./sanity-runner.sh --solution test-sanity --namespace test-sanity-platform --
exec-namespace test-sanity-exec --shared-namespace test-shared-master --ic-
namespace test-sanity-app --ic-client-secret <ic-client-secret> --env-profile
ocp-production --cluster-domain test.sanity --test-option CLEAN
```

Note: After running the CLEAN, if there is a requirement to execute sanity again, there is a need to activate the sanity user in Keycloak prior to re-running.

Note: In this version, all data related to the upgrade sanity script is removed from Minio DB after Minio is restored.

5.2. Enabling Alerts Messaging Via Email

This is an optional feature that enables users to receive alerts messaging via email. Configure as follows:

» To enable alerts messaging via email:

1. In the alerting.yaml.gotmpl:
 - a. Add an email account as a sender with "alert_sender" name. Configure email, host and port.
 - b. Configure email destinations groups or separate emails.

Example:

```
# example for email destination
email_objects:
email_accounts:
- name: alert_sender
```

```
email: sender@youremaildomain.com
host: email.sender.host
port: 587
method: starttls
email_groups:
- name: example_email_group
emails:
- recipient1@email.com
- recipient2@email.com
alerting:
slack:
...
email:
admin_destinations:
- sender: alert_sender
recipients:
email_groups:
- example_email_group
emails:
- recipient3@email.com
regular_destinations:
- sender: alert_sender
recipients:
email_groups:
- example_email_group
emails:
- recipient3@email.com
```

2. Add a name and a password for the email server (get them from your email server provider) in the shared-secrets.yaml:

```
opensearch_alerting_sender_name: ...
opensearch_alerting_sender_pass: ...
encrypt shared secrets if needed.
```

Verify Email destinations via:

Opensearch → Alerting → Destinations.

Note: A test email can be sent in the triggers part of every monitor where these destinations added to a trigger.

5.3. Licensing Procedure Instructions

The section provides the necessary instruction steps to enable users to configure their deployments to run any licensed application components that have been additionally purchased.

5.3.1. Resources

- Online converter - Base64decode
- UUID licenses codes - licenses

Simple string value (e.g. string with license codes)

5.3.2. Procedure Steps

1. Connect to k8s context for apps namespace (in current example uses apps-test namespace)
2. Get all secrets for namespace:

```
oc get secrets -n <apps_namespace>
```

3. Find in list secret with name (e.g. 'tr-icbe-license-secret' - file contains license codes for IC)
4. Edit secret command

```
oc edit secret <file_name> -n <apps_namespace>
```

Example case

```
oc edit secret tr-icbe-license-secret -n apps-test
```

5. Get Base64 encoded string for key 'register.license'

```
# Please edit the object below. Lines beginning with a '#' symbol will be
# ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be
# reopened with the relvant failures.
#
apiVersion: v1
data:
  register.license (property)
  <license code value> {Get property value}
kind: Secret
```

```

metadata:
  annotations:
    helm.sh/hook: pre-install, pre-upgrade
    helm.sh/hook: delete=policy: before-hook-creation
    helm.sh/hook-weight: "-8"
  creationTimestamp: "2023-01=31T07:39:28X"
  labels:
    app.kubernetes.io/instance: tr-icbe
    app.kubernetes.io/managed-by:Helm
    app.kubernetes.io/name: tr-icbe
    app.kubernetes.io/version: 6.0.0
    helm.sh/chart: tr-icbe-0.1.0
  name:tr -icbe-license-secret
  namespace: ievgenba- master- 20230131-app
  resourceVersion:"2109511976"
  uid: a49f9202-11a1-482d-b940-b3b020a1ebc9
type-Opaque

```

6. Decode encoded string.
7. In the editor, add or remove key(s) from decoded string.
8. Encode string into Base64 string.
9. Insert new string as value for key 'register.license' in secret.

```

# Please edit the object below. Lines beginning with a '#' symbol will be
# ignored,
# and an empty file will abort the edit. If an error occurs while saving this
# file will be
# reopened with the relvant failures.
#
apiVersion: v1
data:
  register.license (property)
    <license code value> (placeholder)
kind: Secret
metadata:
  annotations:
    helm.sh/hook: pre-install, pre-upgrade
    helm.sh/hook: delete=policy: before-hook-creation
    helm.sh/hook-weight: "-8"
  creationTimestamp: "2023-01=31T07:39:28X"
  labels:
    app.kubernetes.io/instance: tr-icbe
    app.kubernetes.io/managed-by:Helm
    app.kubernetes.io/name: tr-icbe
    app.kubernetes.io/version: 6.0.0
    helm.sh/chart: tr-icbe-0.1.0

```

```
name:tr-icbe-license-secret
namespace: ievgenba-master-20230131-app
resourceVersion:"2109511976"
uid: a49f9202-11a1-482d-b940-b3b020a1ebc9
type:Opaque
```

10. Save changes.

IMPORTANT: Make sure you get the message that secret is changed

11. Restart pod

- on OCP - restart pod manually
 - get pods from apps namespace
 - delete pod related to tr-icbe (which is in Running status):

```
oc delete pod tr-icbe-xxxxxxxxxx-xxxxx -n <apps_namespace>
```

- wait until pod is deleted and new pod is created

Watch pods in apps namespace until tr-icbe pod will be in Running state (90-120 seconds).

6. Upgrade to 6.11

This section covers:

- Upgrade Preliminary Steps
 - Images
 - Configure Environment Specifics
- Upgrade Steps
 - Run Upgrade

6.1. Upgrade Preliminary Steps

6.1.1. Bastion Server prerequisites

Same prerequisites as required in the installation stage, but be aware that starting from version 6.1 or higher, the helm version used should be 3.7.2 or higher.

6.1.2. Images

Download the ThetaRay installation package to the Bastion server and extract the images tar file:

```
tar xzf tr-platform-images.tgz
tar xzf tr-applications-images.tgz
tar xzf tr-static-images.tgz
```

Load all the images locally with:

```
docker load -i <filename.tar>
```

Tag all the images before pushing to the docker registry:

```
docker tag thetaray/image:tag <docker-registry>/thetaray/image:tag
```

If required, login to the registry with a username and password:

```
docker login <docker-registry>
```

For more information:

<https://docs.docker.com/engine/reference/commandline/login/>

Push the images to the docker registry:

```
docker push <docker-registry>/thetaray/image:tag
```

6.1.3. Configure Environment Specifics

Download and extract ThetaRay helm chart tar files:

replace the x with the actual build number)

```
tar xzf tr-platform-6.*.tgz
tar xzf tr-applications-6.*.tgz
```

extra-values

platform package:

Choose your env profile (e.g. ocp-production) and edit the environment files:

Set cpu and memory requests and limits, volumes sizes and node selector inside:

Note: In case of an upgrade, it is mandatory to align the values in the following two files with the files from the previous release.

- environments/ocp-prod/shared-resources.yaml.gotmpl
- environments/ocp-prod/solution-resources.yaml.gotmpl
- Set docker registry and CA cert if needed:
- environments/ocp-prod/common.yaml.gotmpl
- environments/ocp-prod/backup.yaml.gotmpl

Values:

- image_registry: <registry host-name>
- image_prefix: <registry path>
- image_registry_pull_secret: <a secret to authenticate with the docker registry>
- ca_bundle_secret: <in case of using a self signed certificate or an internal root CA>
- In case of using a specific storage_class, update: environments/ocp-prod/solution-resources.yaml.gotmpl

Values:

- spark_executor:
- storage_class: <storage class>

Values:

- node_selector:
thetaray-env: <disk label>
- node_selector_executors:
thetaray-env: <disk2 label>

Note: 2 node selectors are provided to have the ability to split resources between application and executors, in case you do not need it just put same labels there.

Secrets

Set the passwords of your choice in:

platform package:

- environments/ocp-prod/shared-secrets.yaml.gotmpl
- environments/ocp-prod/solution-secrets.yaml.gotmpl

applications (IC) package:

- helm/values/secrets.yaml

6.1.4. Run Upgrade on Openshift

Note: For the next step it is assumed that the user is working with his own branch, that was created for 6.2.X, for example from Reference, make sure all changes from Jupyter have been pushed.

Note: An upgrade can not be initiated while backup is running.

To verify if a backup is currently running, run:

```
oc get jobs -n <name space>
```

Deleting StatefulSet

For the upgrade, it is recommended to change stateful set values before completing the upgrade stage. Delete the sts (as they are stateless, and cannot be changed)

Related commands:

```
oc delete sts --cascade=orphan postgres-keeper -n <shared-namespace>
oc delete sts --cascade=orphan opensearch-cluster-master -n <shared-namespace>
oc delete sts --cascade=orphan logstash-logstash -n <shared-namespace>
oc delete sts --cascade=orphan minio -n <shared-namespace>
```

Now before the upgrading stage, you need to update the helm, by running the following command.

```
helm uninstall backup -n <shared-namespace>
```

Upgrade Steps:

1.

```
./install-shared.sh --new-shared-namespace <current-shared-namespace> --env-profile ocp-production --cluster-domain <cluster domain> --create-namespace false --customer <customer-bucket-name>
```

Note: <customer-bucket-name> - is a customer identifier, that should be set on installation.

It is used as a part of the backup bucket, to make it unique.

2.

****ONLY FOR GKE****

```
kubectl delete sts --cascade=orphan platform-logstash -n <platform-namespace>
kubectl delete sts --cascade=orphan tr-platform -n <platform-namespace>
kubectl delete sts --cascade=orphan tr-rest-connector -n <platform-namespace>
```

```
./install-env.sh --solution <solution> --namespace <current-platform-namespace> --shared-namespace <current-shared-namespace> --exec-namespace <current-exec-namespace> --env-profile ocp-production --cluster-domain <cluster domain> --create-namespace false
```

If tr-platform-0 Pod fails, then run the following:

```
oc delete pvc solutions-tr-platform-0 -n <platform namespace name>
oc delete pod tr-platform-0 -n <platform namespace name>
```

3.

- a. Edit file values/common-values.yaml in the tr-application. Set the following values:

Note: In case of an upgrade, it is mandatory to align the values in the following file with the file values from the previous release.

- envprofile: "prod"
- platform: "ocp"
- clusterDomain: "<cluster domain>"
- dockerRegistry: <registry host-name>
- dockerRegistryPullSecret: ""
- icInstanceName: "<apps_namespace>"
- sharedNamespaceName: "<infra-namespace>"
- s3BucketName: "<apps-namespace>"
- useWildcardCert: true
- nodeSelector:
 - thetaray-env: <disk label>
- screening:
 - enabled: false

- b. SLA Display in Days

For details on how to change the default SLA (hours to days) refer to Appendix G.

- c. **Note:** In deployments that are aligned with release TR 6.6.1 OR OCP 4.10 and under, you are required to comment out the following code in the related yaml files

```
securityContext:
  capabilities:
    add:
      - NET_BIND_SERVICE
```

This code part is located in two locations:

- applications/helm/environments/ocp-prod/extra-values-ocp-prod.yaml
- applications/helm/environments/ocp-dev/extra-values-ocp-dev.yaml

4.

****ONLY FOR GKE****

```
kubectl delete sts --cascade=orphan tr-logstash -n <apps-namespace>
```

Run the installation:

```
CREATE_NAMESPACE=false helmfile --namespace <current-apps-namespace> -e  
ocp-production -f helmfile-applications.yaml apply
```

Note: Upgrade database does not support sandbox databases that were created by a user with a username that ends with `_ss`.

Now that Upgrade has completed, perform a sanity check according to the instructions located in the [Post-upgrade](#) .

7. Post-upgrade

7.1. Basic Assumptions for Health Check

In order to verify the upgrade passed successfully, execute the Health Check.

Note: A new --test-option **HEALTHCHECK** was added. In the previous version, there were only two options - **run** and **clear**.

Note: From version 6.6, the Healthcheck runs without starting the jupyter hub pod. This functionality reduces costs and improves performance.

7.2. What does the Health Check cover?

- Pods Validation (all relevant pods are alive)
- GitLab- Verification (verification that there are no errors, and branches were created)
- Hasura -Verification that solution exists in deployment
- Minio - Verify: login page is loaded
- jupyterhub - Login to application remotely and test there are no errors
- mlflow - invoke the list of models API and verify that result successfully
- airflow - Invoke API Airflow to receive a list of DAGs
- verify pods are running
- Postgres - validating the Postgres connection by displaying a list of DB tables

The Result of the Sanity test execution will be viewable at the end of the pod automation log.

7.2.1. Running Sanity after upgrade

Note: If an old automation pod is running, delete it before running the Health Check. This is relevant to old automation pods only, and from 6.3 the automation pods will be deleted automatically at every new run.

After shared upgrade run:

```
oc -nSHARED_NAMESPACE logs postgres-keeper-0
```

Check if there is such an error in the logs:

```
ERROR    cmd/keeper.go:743      error getting pg state {"error": "pq:
password authentication failed for user \"replication\""}

```

manual fix:

```
oc -nSHARED_NAMESPACE exec -it postgres-keeper-0 bash
stolon@postgres-keeper-0:/$ psql
postgres=# create user replication with encrypted password 'replication';
postgres=# alter role replication replication;

```

Example:

```
oc -ngk9-shared exec -it postgres-keeper-0 -- bash
stolon@postgres-keeper-0:/$ psql
psql (13.4 (Debian 13.4-4.pgdg110+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256,
compression: off)
Type "help" for help.
postgres=# create user replication with encrypted password 'replication';
CREATE ROLE
postgres=# alter role replication replication;
ALTER ROLE
postgres=#

```

7.2.2. How to run Health Check

Note: If there are problems with logging-in by sanityuser in IC for multi-realm environments, make sure the sanityuser has the following roles "default-roles-applications"

Note: Before sanity after upgrade execute git pull in the reference branch in jupyter.

```
./sanity-runner.sh --solution <solution> --namespace <platform namespace> --  
exec-namespace <exec namespace> --shared-namespace <shared namespace> --ic-  
namespace <ic namespace> --ic-client-secret <ic-client-secret> --env-profile  
ocp-production --cluster-domain <cluster domain> --create-namespace false --  
test-option HEALTHCHECK
```

Example:

```
./sanity-runner.sh --solution test-sanity --namespace test-sanity-platform --  
exec-namespace test-sanity-exec --shared-namespace test-shared-master --ic-  
namespace test-sanity-app --ic-client-secret <ic-client-secret> --env-profile  
ocp-production --cluster-domain test.sanity --create-namespace false --test-  
option HEALTHCHECK
```

Example of successful message:

```

----- Live log Call -----
2022-07-24 11:04:40 [ INFO ] [!] Checked out branch: Reference (service_git.py:189)
2022-07-24 11:04:40,689: INFO:automation.infra_automation.services.service_git:[] Checked out branch: Reference
PASSED

----- Warnings summary -----
.. /.../venv/lib64/python3.8/site-packages/pkg_resources/_init_.py:1144
/thetaray/venv/lib64/python3.8/site-packages/pkg_resources/_init_.py:1144: DeprecationWarning: Use of .. or absolute path in a resource path is not allowed and will raise exceptions in a future release.
  return get_provider(package_or_requirement).get_resource_filename(
.. /.../platform/python/thetaray/apl/solution/risk.py:163
/thetaray/platform/python/thetaray/apl/solution/risk.py:163: DeprecationWarning: Invalid escape sequence \.
  rv.features = [match for (activity,)] for match in re.findall( activity, \w*, rv.expression)]
.. /.../platform/python/thetaray/apl/solution/risk_metadata_repository.py:193
/thetaray/platform/python/thetaray/apl/solution/risk_metadata_repository.py:193: DeprecationWarning: Invalid escape sequence \.
  risk_condition_dict["feature"] = re.search("\.(.*)")[0,1]$", condition.display_settings[0].feature).group(1)
.. /.../platform/python/thetaray/apl/solution/risk_metadata_repository.py:194
/thetaray/platform/python/thetaray/apl/solution/risk_metadata_repository.py:194: DeprecationWarning: Invalid escape sequence \.
  risk_condition_dict["parameter"] = params_dict["parameters"] (re.search("\.(.*)")[0,1]$", condition.display_settings[0].parameter).group(1)
.. /.../platform/python/thetaray/utills/custom_db_inspectors/postgres_single_schema_inspector/queries.py:200
/thetaray/platform/python/thetaray/utills/custom_db_inspectors/postgres_single_schema_inspector/queries.py:200: DeprecationWarning: Invalid escape sequence \:
  INODES = ""
.. /.../venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1043
.. /.../venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1043
.. /.../venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1043
tests/pipeline/test_sanity_health_check.py::test_clone_branches
tests/pipeline/test_sanity_health_check.py::test_remote_execution
/thetaray/venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1043: InsecureRequestWarning: Unverified HTTPS request is being made to host 'keycloak-veronicabe-shared-master-1658408176-3976.ndp.theta
aydev.com'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
tests/pipeline/test_sanity_health_check.py::test_clone_branches
/thetaray/venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1043: InsecureRequestWarning: Unverified HTTPS request is being made to host 'gitlab-veronicabe-shared-master-1658408176-3976.ndp.theta
aydev.com'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
  warnings.warn(
-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== 8 passed, 11 warnings in 322.59s (0:05:22) =====
TEST HEALTHCHECK status is pass =
0
-----
----- TEST HEALTH CHECK Platform/IC PASSED! -----
-----
Finished testing
total 69720

```


Example of failed test message:

```

(R) veronica.berezlavich@ttadmind-Latitude-7400 - /dev/detection-platform
assert len(models_list) == 0, 'MFlow models list is not empty'
else:
    assert len(models_list) > 0, 'MFlow models list is empty'
AssertionError: MFlow models list is empty
E
assert 0 > 0
+ where 0 = len([])

pipeline/test_sanity_health_check.py:122: AssertionError
===== warnings summary =====
../../../../venv/lib64/python3.8/site-packages/pkg_resources/_init_.py:144
  /../../../../venv/lib64/python3.8/site-packages/pkg_resources/_init_.py:144: DeprecationWarning: Use of .. or absolute path in a resource path is not allowed and will raise exceptions in a future release.
    return get_provider(package_or_requirement).get_resource_filename(

../../../../platform/python/thetaray/apl/solution/risk.py:163
  /../../../../platform/python/thetaray/apl/solution/risk.py:163: DeprecationWarning: Invalid escape sequence \.
    rv.features = [match[len('activity.')] for match in re.findall('activity.\w+', rv.expression)]

../../../../platform/python/thetaray/apl/solution/risks_metadata_repository.py:183
  /../../../../platform/python/thetaray/apl/solution/risks_metadata_repository.py:183: DeprecationWarning: Invalid escape sequence \.
    risk_condition_dict['feature'] = re.search('^(.*)$')[0,1]$', condition.display_settings[0].feature).group(1)

../../../../platform/python/thetaray/apl/solution/risks_metadata_repository.py:184
  /../../../../platform/python/thetaray/apl/solution/risks_metadata_repository.py:184: DeprecationWarning: Invalid escape sequence \.
    risk_condition_dict['parameter'] = params_dict['parameters'][re.search('^(.*)$')[0,1]$', condition.display_settings[0].parameter).group(1)]

../../../../platform/python/thetaray/utills/custom_db_inspectors/postgres_single_schema_inspector/queries.py:208
  /../../../../platform/python/thetaray/utills/custom_db_inspectors/postgres_single_schema_inspector/queries.py:208: DeprecationWarning: Invalid escape sequence \.
    INDEXES = """

../../../../venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1843
  /../../../../venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1843: InsecureRequestWarning: Unverified HTTPS request is being made to host 'keycloak-veronicabe-shared-master-1658488176-3976.ndp.thetaray.com'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
    warnings.warn(

tests/pipeline/test_sanity_health_check.py::test_clone_branches
  /../../../../venv/lib64/python3.8/site-packages/urllib3/connectionpool.py:1843: InsecureRequestWarning: Unverified HTTPS request is being made to host 'gitlab-veronicabe-shared-master-1658488176-3976.ndp.thetaray.com'. Adding certificate verification is strongly advised. See: https://urllib3.readthedocs.io/en/1.26.x/advanced-usage.html#ssl-warnings
    warnings.warn(

-- Docs: https://docs.pytest.org/en/stable/warnings.html
===== short test summary info =====
FAILED pipeline/test_sanity_health_check.py::test_mflow_model_validation - A...
===== 1 failed, 3 passed, 4 skipped, 10 warnings in 345.28s (0:05:45) =====
TEST HEALTHCHECK status is pass = 1
----- TEST HEALTH CHECK Platform/IC FAILED! -----
Finished testing
Total 69692
0 success 1 root root 6969 30 24 10:23

```

7.3. Licensing Procedure Instructions

The section provides the necessary instruction steps to enable users to configure their deployments to run any licensed application components that have been additionally purchased.

7.3.1. Resources

- Online converter - Base64decode
- UUID licenses codes - licenses

Simple string value (e.g. string with license codes)

7.3.2. Procedure Steps

1. Connect to k8s context for apps namespace (in current example uses apps-test namespace)
2. Get all secrets for namespace:

```
oc get secrets -n <apps_namespace>
```

3. Find in list secret with name (e.g. 'tr-icbe-license-secret' - file contains license codes for IC)
4. Edit secret command

```
oc edit secret <file_name> -n <apps_namespace>
```

Example case

```
oc edit secret tr-icbe-license-secret -n apps-test
```

5. Get Base64 encoded string for key 'register.license'

```
# Please edit the object below. Lines beginning with a '#' symbol will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving this
file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  register.license (property)
  <license code value> {Get property value}
kind: Secret
metadata:
  annotations:
    helm.sh/hook: pre-install, pre-upgrade
    helm.sh/hook: delete=policy: before-hook-creation
    helm.sh/hook-weight: "-8"
  creationTimestamp: "2023-01-31T07:39:28X"
  labels:
    app.kubernetes.io/instance: tr-icbe
    app.kubernetes.io/managed-by: Helm
    app.kubernetes.io/name: tr-icbe
    app.kubernetes.io/version: 6.0.0
    helm.sh/chart: tr-icbe-0.1.0
  name: tr-icbe-license-secret
  namespace: ievgenba-master-20230131-app
  resourceVersion: "2109511976"
  uid: a49f9202-11a1-482d-b940-b3b020a1ebc9
type: Opaque
```

6. Decode encoded string.
7. In the editor, add or remove key(s) from decoded string.
8. Encode string into Base64 string.
9. Insert new string as value for key 'register.license' in secret.

```
# Please edit the object below. Lines beginning with a '#' symbol will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving this
```

```

file will be
# reopened with the relvant failures.
#
apiVersion: v1
data:
  register.license (property)
    <license code value> (placeholder)
kind: Secret
metadata:
  annotations:
    helm.sh/hook: pre-install, pre-upgrade
    helm.sh/hook: delete=policy: before-hook-creation
    helm.sh/hook-weight: "-8"
  creationTimestamp: "2023-01=31T07:39:28X"
  labels:
    app.kubernetes.io/instance: tr-icbe
    app.kubernetes.io/managed-by:Helm
    app.kubernetes.io/name: tr-icbe
    app.kubernetes.io/version: 6.0.0
    helm.sh/chart: tr-icbe-0.1.0
  name:tr -icbe-license-secret
  namespace: ievgenba- master- 20230131-app
  resourceVersion:"2109511976"
  uid: a49f9202-11a1-482d-b940-b3b020a1ebc9
type:Opaque

```

10. Save changes.

IMPORTANT: Make sure you get the message that secret is changed

11. Restart pod

- on OCP - restart pod manually
 - get pods from apps namespace
 - delete pod related to tr-icbe (which is in Running status):

```
oc delete pod tr-icbe-xxxxxxxxxx-xxxxx -n <apps_namespace>
```

- wait until pod is deleted and new pod is created

Watch pods in apps namespace until tr-icbe pod will be in Running state (90-120 seconds).

8. Appendix A - Data Tiering

8.1. Introduction

The ThetaRay system leverages Object Storage based on Minio which is deployed into a Kubernetes / Openshift environment and utilizes by default local SSDs for persistent data storage. Public cloud environments (for this release Microsoft Azure is supported) provide managed object storage facilities which provides a more cost effective way for storing data. Data Tiering is a functionality that enables data to be dynamically moved from local disks to remote object storage based on user configurable policies, while still being able to access the data, as if it was locally stored in Minio.

In order to enable Data Tiering, please adhere to the following prerequisites and installation instructions:

8.2. Azure Prerequisites

A prerequisite of using data tiering is to:

1. Create two storage accounts in azure for the Hot and Warm tier as follows:
 - a. The Hot tier should have "performance" set as Premium while the Warm tier should have "performance" set as Standard.
 - b. The other storage account attributes are for installer consideration. We recommend enabling Zone-redundant Storage (ZRS) replication and Infrastructure Encryption.
2. Create a container in each storage account.
3. Optionally, create Service Principal(s) in Azure AD / Entra and grant access to the Azure BLOB Containers containing the 'tiered' data. The client id, client secret and tenant id associated with these Service Principals can be used as an alternative authentication mechanism to the default 'access key' based authentication.

8.3. Deployment Parameters

In `common.yaml.gotmpl`:

```
data_tiers:
  azure_store_region: <Region of the storage account>
  datatiers_enabled: true
  datatiers_enable_hot_tier: true
```

```
datatiers_enable_warm_tier: true
datatiers_hot_account: <Hot tier storage account>
datatiers_hot_bucket: <Hot tier container>
datatiers_warm_account: <Warm tier storage account>
datatiers_warm_bucket: <Warm tier container>
```

In case Access Key based authentication is used to access Azure BLOB, in **shared-secrets.yaml** -

```
hot_store_access_key: <Hot storage account access key>
warm_store_access_key:<Warm storage account access key>
```

In case Service Principal based authentication is used to access Azure BLOB, **shared-secrets.yaml** should contain the Service Principal credentials as detailed below -

```
warm_client_id: <client id>
warm_client_secret: <secret>
warm_tenant_id: <tenant id >
hot_client_id: <client id>
hot_client_secret: <secret>
hot_tenant_id: <tenant id>
```

9. Appendix B - Data at Rest Encryption

Data at rest can optionally be encrypted at application level. The encryption covers – raw input files, specific columns consisting of sensitive information in Postgres and Minio, PII fields in Opensearch and exported audit messages.

To enable data at rest encryption ‘—data-encryption true’ should be passed as a command line argument to the ‘install-shared.sh’ and the ‘install-env.sh’ scripts when installing the shared infrastructure and platform instance. To enable data at rest encryption within the applications namespaces – secrets.yaml should be edited using ‘sops’ and ‘icEncryptionDecryption’ should be set to an ‘enabled’ state.

To facilitate the encryption process a ‘master’ asymmetric (RSA 2048) Key Encryption Key should be set up within Azure Key Vault and be made accessible to the ThetaRay environment for issuing ‘Wrap Key’ / ‘Unwrap Key’ operations. These operations are used to ensure the security of Data Encryption Keys established and maintained within the ThetaRay environment.

Establishing an Azure Key Vault instance and securing it should be handled by the customer. Azure Key Vault connection information should be set up in the ‘shared-secrets.yaml’ environment configuration file (since the file is encrypted ‘sops’ should be used to edit it), with the following settings required to be provided

- uri – the Azure Key Vault URI - kek_name – The name of the Key within Azure Key Vault that holds the Key Encryption Key
- tenant_id – Azure AD tenant used for authentication
- client_id – Azure AD client / application registration identifier used for access
- either a client_secret or a certificate + password used as credentials

Following is an example snippet from shared-secrets –

azure_key_vault:

- uri: <key vault URI>
- kek_name: <key name>
- tenant_id: <tenant id>
- client_id: <client id>

secret:

- client_secret: null

certificate:

- certificate_password: <certificate password>

- azure_ad_key: <base64 encoded certificate>

9.1. Using Customer Provided Data Encryption Key (DEK)

By default , when enabling data at rest encryption on the environment, a random data encryption key is generated by the deployment process and is registered within the environment. This process implies that each environment is assigned with unique data encryption keys, preventing files encrypted on one environment from being decrypted on another.

In some cases, such as having a pair of environments consisting of production data - one serving as the actual production and the other as a simulation / calibration environment, ongoing data transfer across environment is part of the BAU process requiring encrypted data originating from one environment to be decrypted on another. To facilitate this process, the deployment configuration can be setup to use a customer provided DEK instead of randomly generating one. This allows the same DEK to be re-used across environments, enabling sharing of data.

➤ **To configure the DEK - use SOPS to edit shared-secrets and set ->**

data_encryption:

dek:

pii: <64 characters random hex string used for column level encryption>

audit: <64 characters random hex string used for audit messages export encryption>

10. Appendix A - Backup and Disaster Recovery

10.1. Introduction

To be able to ensure system and data availability in scenarios such as whole data center (region) unavailability for a significant amount of time or data corruption due to technical or user errors the system provides the following capabilities:

- Periodic, scheduled data backup to external storage - a process that backs up the data managed by the system to a storage device running outside of the Kubernetes / Openshift environment. Backups are used to protect against data corruption due to human error or technical failures.

Currently the following backup targets are supported:

- Azure BLOB Storage - for environments running on Openshift / Azure
- Google Cloud Storage - for environments running on Google Kubernetes Engine
- S3 Compatible Storage - for on-prem environments
- Continuous data replication from an active environment on a primary data center / region to a passive, fail over site. The mechanism enables starting an environment on a fail-over site using replicated data, and requires a running Minio cluster on the remote site to capture and persist replicated data (Minio running over Kubernetes / Openshift).

The mechanism is agnostic to the underlying infrastructure and works on both on-prem / cloud environments with the only assumption being is the availability of network connectivity between the sites.

The replication facility performs continuous replication of both Postgres and Minio data across data centers. Replication of monitoring information in Opensearch and data within the Gitlab repository are replicated using daily snapshots.

The following diagram shows Backup and Disaster Recovery in a high level.

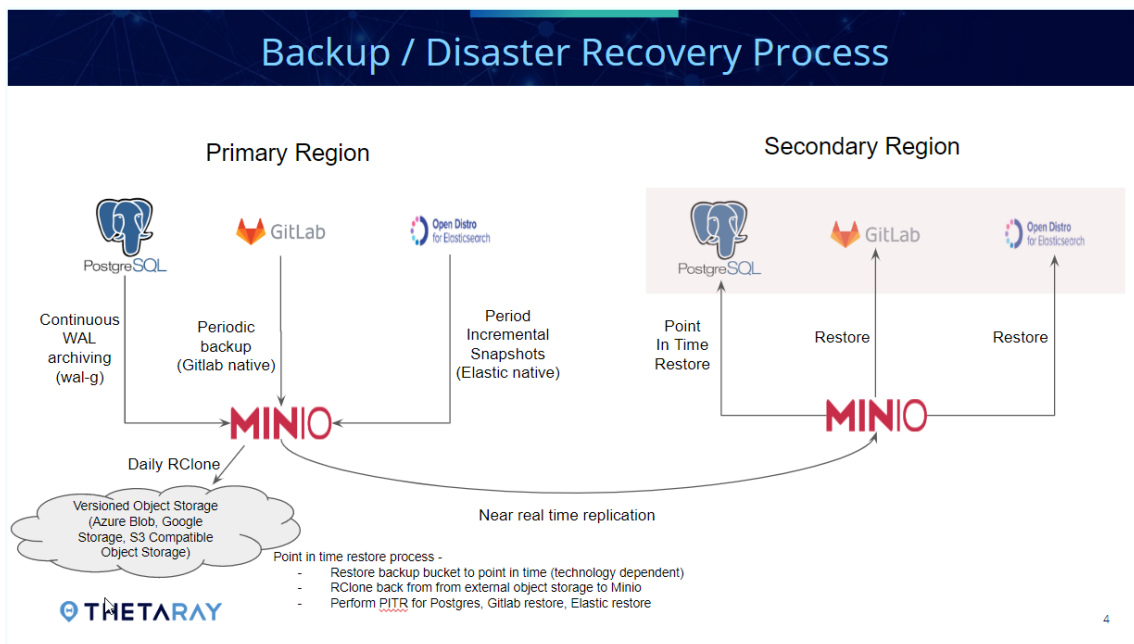


Figure 3: Over view of the Backup and Disaster Recovery Architecture

10.2. Backup, Restore and Disaster Recovery

10.2.1. Purpose and Scope

The purpose and scope of this document is to provide you with a comprehensive description and details of how to run backup, restore and disaster recovery when working with your ThetaRay deployment.

Also be aware, that elements of this process, specifically related to setting up backup components are also covered in the current version of the ThetaRay Installation and setup guide. Personnel, primarily tasked with system deployment may find it a more efficient use of their time, to refer directly to this document for specific back up configuration instructions.

10.2.2. Overview

Thetaray offers backup and disaster recovery services out of box.

The solution proposed aims at solving the two related categories of back up:

Data backups - the ability to perform periodic (daily) backups of data managed by the system to external storage, allowing the system to be restored to a given point in time.

Disaster Recovery - a mechanism that enables ongoing replication of data managed by the system to an secondary data center, allowing the system to be recovered in case of a disaster within the primary data center.

10.2.3. Backup

10.2.4. Prerequisites for Setting up Backup Targets

- Backup target is an external Object Storage that provides object level versioning allowing point in time restore of the system.
 - Google Storage
 - Azure Blob Storage
 - S3 Compatible Storage

Note: Throughout this document, when performing copy paste of commands, pay attention to maintaining correct syntax.

10.2.4.1. Google Cloud Storage

[For detailed documentation, refer to <https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity>]

» **To setup backup to Google cloud storage data, proceed with the following steps:**

1. Enable workload on your cluster.

```
gcloud container clusters update CLUSTER_NAME \
  --region=COMPUTE_REGION \
  --workload-pool=PROJECT_ID.svc.id.goog
```

2. Create/update nodepool with flag: --workload-metadata=GKE_METADATA:

```
gcloud container node-pools update NODEPOOL_NAME \
  --cluster=CLUSTER_REGIONNAME \
  --workload-poolmetadata=GKE_METADATA
```

3. Create a kubernetes service account:

```
kubectl create serviceaccount KSA_NAME --namespace NAMESPACE
```

4. Create a Google cloud service account provide an id and with read/write access permission to the required bucket repository:

```
gcloud iam service-accounts create GSA_NAME --project=GSA_PROJECT
gcloud projects add-iam-policy-binding PROJECT_ID \
  --member "serviceAccount:GSA_NAME@GSA_
PROJECT.iam.gserviceaccount.com" \
  --role "roles/storage.admin"
```

Optional: You can also create your custom role with access to the specific bucket where you are going to write backups. Permissions needed: list buckets, create bucket, admin permissions on this bucket.

5. Connect the kubernetes service account with a Google cloud service account:

```
kubectl annotate serviceaccount KSA_NAME \
  --namespace NAMESPACE \
  iam.gke.io/gcp-service-account=GSA_NAME@GSA_
PROJECT.iam.gserviceaccount.com
```

6. Annotate the kubernetes service account with a pointer to the Google cloud service account:

```
kubectl annotate serviceaccount KSA_NAME \
  --namespace NAMESPACE \
  iam.gke.io/gcp-service-account=GSA_NAME@GSA_
PROJECT.iam.gserviceaccount.com
```

10.2.4.2. Azure Blob Storage

» To setup backup to Azure blob storage data, proceed with the following steps:

1. Create an azure storage account:

```
az storage account create --name {CUSTOMER NAME}drsa --resource-group
{CUSTOMER RG} --subscription {SUBSCRIPTION}
```

2. Enable versioning on this storage account:

```
az storage account blob-service-properties update \
  --resource-group {CUSTOMER RG} \
  --account-name {CUSTOMER NAME}drsa \
  --enable-delete-retention true \
  --delete-retention-days 30 \
  --enable-versioning true \
```

```
--enable-change-feed true \
--enable-restore-policy true \
--restore-days 29 \
--subscription {SUBSCRIPTION}
```

*restore-days < retention-days

**restore-days is number of days that allows possibility to perform point in time restore

3. Generate shared access URI

```
az storage account generate-sas \
--account-name {CUSTOMER NAME}drsa \
--expiry 2030-01-01 \
--resource-types sco \
--permissions rwdacul \
--services bf \
--subscription {SUBSCRIPTION}
```

4. Grab the output and go to the **shared-secrets.yaml, encrypt** token you received, update **shared_access_token** with encrypted value. You can achieve that by using **tools/sops/sops_cmd.sh** to decrypt the file, add raw **shared_access_token**, and encrypt the file back using **tools/sops/sops_cmd.sh** again.
5. Go to **backup.yaml.gotmpl** and set **storage_account_name** to your customer storage account name.

As an alternative to Shared Access Signature, the system supports authentication through Azure Service Principal. To enable this type of authentication, a service principal with the necessary access permissions to the Storage Account should be registered within Azure AD and Entra. The Tenant ID / Client ID / Client Secret associated with the Service Principal should be setup in the backup section within **shared-secrets.yaml** as detailed later in the document.

10.2.5. S3 Compatible Storage

ThetaRay provides the ability to backup to S3 compatible storage with different providers using the Rclone tool [<https://rclone.org/>].

1. Basic procedure:
 - a. Configure a connection to S3 compatible storage, as is presently done with Rclone.

The list of supported providers and needed configuration can be found here - <https://rclone.org/s3/>.

- If you use any another provider and faced any problems contact Support.
- Regarding configuration there are basic settings that are required for all S3 providers:
 - b. endpoint
 - c. access_key_id
 - d. secret_access_key
- This settings is sufficient for most of open-source S3 providers.
- If there is situation when you need any additional settings, like acl or location:
 - e. Provide it through deployment configuration in section backup.s3.extra_config. This config will be parsed and included to rclone config.

10.2.6. Configuration & Installation

10.2.6.1. Configuration

The configuration file is represented as an environmental values YAML file and is split into sections according to system components.

See more about values configuring in the specific component section. Here you can find a short overview of each configuration value.

Inner configuration of components like bucket creation, setup, or number backups of retain are made on backup release installation and rely on config values.

backup.yaml.gotmpl

```
backup:
  enabled: true      # enable or disable automated backups
  schedule: "0 0 * * *" # backuping schedule for Opensearch, Gitlab,
  incremental PostgreSQL backup and backup to external storages;
  postgres:
    number_to_retain: 1 # number of full backups to retain
```

```

    bucket: pg-archive          # MinIO bucket to keep PostgreSQL backups
    differential_backup_days_interval: 7          # Interval of days to do
differential backup in
    full_backup_days_interval: 30          # Interval of days to do full
backup in
    pit_to_restore: "{{ env \"PG_PIT_TO_RESTORE\" }}"          # Time to restore
in postgres, set through install-shared script
    gcp:
        enabled: false          # enable/disable backups of MinIO to Google Cloud
Storage
        versions_retain_days: 30 # number of days to keep objects version at
google storage
        bucket: minio-backups-{{ requiredEnv \"CUSTOMER\" }}-{{ requiredEnv
\"NDP_SHARED_NAMESPACE\" }}          # Google Cloud Storage bucket to save
MinIO backups
        k8_iam_service_account: gcp-backup          # Kubernetes service
account that is connected to gcp with proper permission read/write at
Google Cloud Storage
        azure:
            enabled: false          # enable/disable backups of MinIO to Azure Blob
Storage
            container: minio-backups-{{ requiredEnv \"CUSTOMER\" }}-{{ requiredEnv
\"NDP_SHARED_NAMESPACE\" }}          # Azure Blob Storage container to save
MinIO backups
            shared_access_token: \"token\"          # shared access token to Azure
Blob Storage
            storage_account_name: drtestsa          # Azure Blob Storage account name
        s3:
            enabled: false          # enable/disable backups of MinIO to S3 Storage
            versions_retain_days: 30 # number of days to keep objects version
at google storage
            bucket: minio-backups-{{ requiredEnv \"CUSTOMER\" }}-{{ requiredEnv
\"NDP_SHARED_NAMESPACE\" }}          # S3 Storage bucket to save MinIO backups
            endpoint: s3_endpoint # endpoint to connect to S3 storage
            access_key_id: thetaray          # access key
            secret_access_key: thetaray          # secret key
            provider: Minio          # Provider of S3 storage. We're provider should
be set according to rclone configuration. More details in appropriate
documentaion chapter.
            # extra_config:          # Extra configuration needed for specific
provider to setup rclone config. More details in appropriate documentaion
chapter.
            #   acl: private
            #   location_constraint: us-east-1
        gitlab:
            number_to_retain: 5          # number of GitLab backups to retain
            bucket: gitlab-backups-storage          # MinIO bucket to keep GitLab
backups

```

```
opensearch:
  bucket: oss-archive      # MinIO bucket to keep Opensearch backups
```

10.2.6.2. Installation

After the values have been configured, you should run the shared environment installation command as usual:

```
./install-shared.sh --new-shared-namespace { namespace_name } --env-
profile gke-development --cluster-domain { cluster_domain } --customer
customer
```

» How to check if installation is fine:

1. If you configured backups to external storage (Google Cloud Storage and Azure Blob Storage), you should check to pre-install the job that is responsible for the creation of the backup bucket or check if it exists.

To do this, list the pods and check for completion of relevant job:

```
kubectl -n { ns } get po
```

NAME	READY	STATUS	RESTART
S AGE			
....			
# if Azure Blob Storage configured			
backup-create-azure-bucket-			
qggzj 0/1 Completed 0	27m		
....			
# if Google Cloud Storage configured			
backup-create-gcp-bucket-			
qggzj 0/1 Completed 0	27m		
....			
# if S3 Storage configured			
backup-create-s3-bucket-			
qggzj 0/1 Completed 0	27m		

2. If these jobs are completed successfully, you should check the list of your cronjobs to be sure that the backup release was deployed correctly.

According to the list of cron jobs in your namespace:

```
kubectl -n { ns } get cj
```

NAME	SCHEDULE	SUSPEND	ACTIVE	LAST
SCHEDULE AGE				
backup 0 0 * *				
* False 0 <none> 1m				

```
gitlab-task-runner-backup 0 0 * *
* False 0 <none> 1m
# And this job should appears if you configure backups to external
storage
backup-to-external-storage 0 0 * *
* False 0 <none> 1m
```

10.2.7. Backup Running Process Overview

Backups are run automatically with cron jobs relying on the configuration provided through the values. You can view cronjob examples that will be run during the backup:

```
backup # Job responsible for incremental/differential/full PostgreSQL
backup and OpenSearch backup
gitlab-task-runner-backup # Job responsible for GitLab backup
backup-to-external-storage # Job responsible for backup to external
storage
```

Important. As was shown in the configuration section cronjobs "backup", "gitlab-task-runner-backup" and "backup-to-external-storage" use the same schedules to run cronjobs. But it's not running at the same time but one by one. Firstly "gitlab-task-runner-backup" that executing the backup of GitLab then the "backup" job starts responsible for PostgreSQL and OpenSearch backups and finally the last one is "backup-to-external-storage".

Also, the ability to run manual backups with scripts is included. Details about running backups manually in this way are provided in the section "Manual backups"

10.2.8. Backup Components Overview

By default, backup content and process includes:

1. Postgresql - continuous archiving & daily incremental backup & weekly differential backup & monthly full backup to Minio.
2. Opensearch - daily incremental backup to Minio.
3. Gitlab daily full backup to Minio.
4. Minio backups to external storage - Azure Blob Storage or Google Cloud Storage.

10.2.8.1. PostgreSQL

The Postgres backup is based on the 'pgbackrest' tool [https://pgbackrest.org/] that maintains archived Write Ahead Logs, Full and differential (changes since last full backup) backups in a MinIO bucket.

From version 6.2, all types of backup are made in single job - backup and the type of backup is calculated on the backup interval set in configuration.

You can check the status of performed backup with a pod list where you can find completed pods.

```
kubectl -n { ns } get po
```

NAME	READY	STATUS	RESTART
S AGE			
backup-1652832000-mzv9s	0/1	Completed	0 7h7m

```

kubectl -n arturva-shared-master-20220328-1698 logs -f backup-1652572800-2tbd8
Check if there is no another backup in progress
....
Executing backup of type {diff/full/incr}.
...
Creation of diff backup postgresql finished

```

Sample of PostgreSQL configuration section:

```

schedule "0 0 * * *"
postgres:
  number_to_retain: 1
  bucket: pg-archive
  differential_backup_days_interval: 7
  full_backup_days_interval: 30
  pit_to_restore: "{{ env \"PG_PIT_TO_RESTORE\" }}"

```

» To view 6 config values:

1. schedule - this is a general schedule for backups made for all components including incremental Postgres backup. The recommended value is "0 * * * *" (daily)
2. number_to_retain - This value defines the number of last FULL backups to keep in MinIO bucket. The recommended value is 1 to not accumulate a large number of full backups of postgres database.

3. bucket - the name of the bucket that will be created on installation where all backups of Postgres will be kept.
4. differential_backup_days_interval - Interval of days to do differential backup in. The recommended value is 7 (weekly). If the last backup is full and all other backups where expired the count will be taken from last full backup. It means that next differential backup will be made in 7 days after last full backup.
5. full_backup_days_interval - Interval of days to do full backup in. The recommended value is 30 (monthly).
6. pit_to_restore - point in time to restore postgres, is set through install-shared.sh script.

Backup types explanation:

- wal-logs - archived Write Ahead Logs that are written to MinIO after each operation.
- incremental - incremental from the last successful backup. On execution of an incremental backup all previous WALs are deleted.
- differential - like an incremental backup but always based on the last full backup. On execution of a differential backup all previous incremental backups are deleted.
- full - all database cluster files will be copied and there will be no dependencies on previous backups. On execution of a full backup all previous differential backups are deleted.

10.2.8.2. OpenSearch

OpenSearch backups are based on built-in snapshots. Snapshots are backups of a cluster's indices and state. State includes cluster settings, node information, index metadata (mappings, settings, templates, etc.), and shard allocation. OpenSearch snapshots are incremental, meaning that they only store data that has changed since the last successful snapshot. The difference in disk usage between frequent and infrequent snapshots is often minimal.

Snapshots are triggered by the "backup" cron job through OpenSearch REST API and saved to a MinIO bucket. Backups are registered to OpenSearch repository named "oss-archive". A snapshot repository is just a storage location, MinIO in our case. You can discover there, an OpenSearch backups list. Backups are named with the date when created.

```
GET _snapshot/oss-archive
{
  "oss-archive" : {
```

```

        "type" : "s3",
        "settings" : {
          "bucket" : "oss-archive"
        }
      }
    }
  }

GET _snapshot/oss-archive/_all
{
  "snapshot" : "2022_05_18_00_04_13",
  "uuid" : "SKpHB9fiQlupauv4cdjd_A",
  "version_id" : 135238127,
  "version" : "1.2.3",
  "indices" : [
    ".kibana_-1067363439_tradmin_1",
    "events",
    ".opendistro-anomaly-results-history-2022.05.17-1",
    ".kibana_1",
    "metrics_artur17",
    "events_artur17",
    ".opendistro-anomaly-detector-jobs",
    ".opendistro-alerting-config",
    ".opendistro-job-scheduler-lock",
    ".opendistro-anomaly-detection-state",
    "security-auditlog-2022.05.17",
    ".opendistro-alerting-alerts",
    ".opendistro-alerting-alert-history-2022.05.17-1",
    ".opendistro-anomaly-checkpoints",
    ".opendistro-anomaly-detectors"
  ],
  "data_streams" : [ ],
  "include_global_state" : true,
  "state" : "SUCCESS",
  "start_time" : "2022-05-18T00:04:13.453Z",
  "start_time_in_millis" : 1652832253453,
  "end_time" : "2022-05-18T00:04:16.655Z",
  "end_time_in_millis" : 1652832256655,
  "duration_in_millis" : 3202,
  "failures" : [ ],
  "shards" : {
    "total" : 15,
    "failed" : 0,
    "successful" : 15
  }
}

```

Sample of OpenSearch configuration section

```
schedule "0 * * * *"
opensearch:
  bucket: oss-archive
```

» To view 2 config values:

1. schedule - this is a general schedule for backups made for all components including incremental OpenSearch backup. The recommended value is "0 * * * *" (daily)
2. bucket - the name of the bucket that will be created on installation, where all backups of OpenSearch will be kept.

You can easily check the status of performed backups by viewing the pod list where you can find completed pods:

```
kubectl -n { ns } get po
NAME                                READY   STATUS    RESTART
S   AGE
backup-1652832000-
mzv9s                                0/1     Completed 0          7h7m
....

kubectl -n arturva-shared-master-20220328-1698 logs -f backup-1652832000-
mzv9s
....
Backup to opensearch started
Creating of opensearch backup: 2022_05_23_00_03_39 started
Creating of opensearch backup: 2022_05_23_00_03_39 finished
Backup to opensearch finished
```

10.2.8.3. GitLab

The Gitlab backup is performed through GitLab's built-in backup facility and performs full backups of the state of the component into the configuration MinIO bucket. The backup job runs by GitLab task runner with "gitlab-task-runner-backup" cronjob

Sample of GitLab configuration section

```
schedule "0 0 * * *"
gitlab:
  number_to_retain: 5
  bucket: gitlab-backups-storage
```

» To view 3 config values:

1. `schedule` - this is a general schedule for backups made for all components GitLab backup. The recommended value is `"0 * * * *"` (daily).
2. `number_to_retain` - as the GitLab built-in utility makes only full backups we don't want to keep a lot of them so here is an option to limit the number of backups.
3. `bucket` - the name of the bucket that will be created on installation where all backups of GitLab will be kept.

You can easily check the status of performed backups by viewing the pod list where you can find completed pods:

```
kubectl -n { ns } get po
NAME                                READY   STATUS    RESTART
S   AGE
gitlab-task-runner-backup-27554400-
nmqv5    0/1     Completed  0          9h

kubectl -n arturva-shared-master-20220328-1698 logs -f gitlab-task-
runner-backup-27554400-nmqv5
....
Packing up backup tar
WARNING: Module python-magic is not available. Guessing MIME types based
on file extensions.
[DONE] Backup can be found at s3://gitlab-backups-storage/1653264033_
2022_05_23_14.3.2_gitlab_backup.tar
Found 6 existing backups. Maximum allowed is 5
Deleting old backup s3://gitlab-backups-storage/1652832010_2022_05_18_
14.3.2_gitlab_backup.tar
[DONE] Finished pruning old backups
```

10.2.9. Backup to External Storage

Before enabling backups to external storage either Google Cloud Storage or Azure Blob Storage, be sure that you configured as described above in the section, "Prerequisites for setting up backup targets."

Backup to external is a backup of MinIO to cloud storage outside of the cluster, to ensure it is maintained in a place of maximum safety regarding the occurrence of possible system issues.

Backup of MinIO means that all non-empty buckets will be copied to external storage, here is an approximate list of buckets that will be copied:

- All platform solution buckets "thetaray-{ solution }", "thetaray-public-{ solution }"
- All IC buckets "{ solution }-app"
- PostgreSQL backup bucket (name provided via config)
- OpenSearch backup bucket (name provided via config)
- GitLab backup bucket (name provided via config)
- GitLab required buckets: "gitlab-tmp-storage", "gitlab-lfs-storage", "gitlab-uploads-storage"

The backup job is run by the "backup-to-external-storage" cronjob.

Backups are made with the rclone tool [<https://rclone.org/>]. Rclone is a command-line program to manage files on cloud storage.

Sample of backup to Google Cloud Storage and Azure Blob Storage configuration section:

```

schedule "0 0 * * *"
gcp:
  enabled: false
  versions_retain_days: 30
  bucket: minio-backups-{{ requiredEnv "NDP_SHARED_NAMESPACE" }}
  k8_iam_service_account: gcp-backup
azure:
  enabled: false
  container: minio-backups-{{ requiredEnv "NDP_SHARED_NAMESPACE" }}
  shared_access_token: "token"
  storage_account_name: drtestsa
  tenant_id: <optional - service principal tenant id>
  client_id: <optional - service principal client id>
  client_secret: <optional : service principal client secret>
s3:
  enabled: false
  versions_retain_days: 30
  bucket: minio-backups-{{ requiredEnv "NDP_SHARED_NAMESPACE" }}
  endpoint: s3_endpoint
  access_key_id: thetaray
  secret_access_key: thetaray
  provider: Minio
  # extra_config:
  #   acl: private
  #   location_constraint: us-east-1

```

>> To view 2 config values:

1. schedule - this is a general schedule for backups made for all components including backup of MinIO to external storage. The recommended value is "0 * * * *" (daily). Important: this job waits for all other components to finish with their backups and only after they finish it starting copy it to external storage.
2. gcp - section to configure backups to Google Cloud Storage.
 - a. enabled - flag that indicates if you need to do backups to Google Cloud Storage.
 - b. versions_retain_days - number of days to keep object versions to perform pitr on it. Configured automatically
 - c. bucket - the name of the bucket to keep backups.
 - d. k8_iam_service_account - the name of Kubernetes service account that was configured and granted access to use workload identity of google service account inside of a pod. Please see "Prerequisites for setting up backup targets" to check how to configure it.

3. azure - section to configure backup to Azure Blob Storage.
 - a. enabled - flag that indicates if you need to do backups to Azure Blob Storage.
 - b. container - basically it's the same as a bucket regarding where to keep backups.
 - c. shared_access_token - shared access token to Azure Blob Storage. Please see "Prerequisites for setting up backup targets" to check how to configure it.
 - d. storage_account_name - account name for which shared access token and URI were issued. Please see "Prerequisites for setting up backup targets" to check how to configure it.
 - e. storage_account_name: sdfsf - tenant_id: <optional - service principal tenant id>

client_id: <optional - service principal client id>

4. s3 - section to configure backup S3 Storage:
 - a. enabled - flag that indicates if you need to do backups to S3 Storage
 - b. versions_retain_days - number of days to keep object versions to perform pitr on it. Is applied only if provider = Minio.
 - c. bucket - the name of the bucket to keep backups.
 - d. endpoint - URL to S3 Storage API. Be aware it can be different from UI URL. For example for minio you should provide https://minio..... and not https://minio-console....
 - e. access_key_id - S3 access key
 - f. secret_access_key - S3 secret Key
 - g. provider - S3 Provide, if you're using not Minio S3 storage please check S3 Storage prerequisites
 - h. extra_config - additional configuration for S3 storage required by Rclone, if you're using not Minio S3 storage please check S3 Storage prerequisites

You can easily check the status of performed backups by viewing the pod list where you can find completed pods:

```
kubectl -n { ns } get po
```

NAME	READY	STATUS	REST
ARTS	AGE		
...			
backup-to-external-storage-27554400-			

lfsx8	0/1	Completed	0	9h
...				

10.2.10. Manual Backups

Also in the ThetaRay backup system, scripts are provided to do manual backups. They are located in the backup-recovery folder.

Important: You can perform manual backups only if backups infrastructure were configured with your shared environment instance on installation.

```
./backup-tool.sh
Usage: ./backup-tool.sh [options]

Multitool script to manage backups and restores

Available commands:
  external          Manage backup and restore from external
storage
  platform          Manage manual backups of platform
components
  promote-failover-site  Restore backups on recovered environment
  setup-replication    Setup replication between 2 MinIO instances

  --help            Help

Usage example:
./backup-tool.sh [command] [subcommands] --help
./backup-tool.sh promote-failover-site --help
./backup-tool.sh platform backup-postgresql --help
./backup-tool.sh external s3 restore --help
```

10.2.10.1. Internal Platform Backups

1. PostgreSQL manual backup script.

Manual base backup of PostgreSQL

```
./backup-tool.sh platform backup-postgresql --help
# Usage: ./backup-tool.sh platform backup-postgresql [options]
#
# Options:
#   --shared-namespace      Shared namespace
#   --type                  Type of backup to do. Allowed values: ["full",
"diff", "incr"].
#   --debug                Set to true if want run
postgresql backup in debug mode
#   --help                  Help

# Example
./backup-tool.sh platform backup-postgresql --shared-namespace arturva-
shared-master-20220117-77 --type full

Backing started
2022-06-29 19:59:10.241 P00  INFO: backup command begin 2.39: --delta --
```

```
exclude=postgresql.auto.conf --exec-id=41290-23c4447f --io-timeout=900 --
log-level-console=info --log-level-file=debug --log-path=/tmp --pg1-
path=/stolon-data/postgres --process-max=4 --repo1-bundle --repo1-bundle-
limit=10MiB --repo1-bundle-size=30MiB --repo1-path=/primary --repo1-
retention-full=1 --repo1-s3-bucket=pg-archive --repo1-s3-
endpoint=minio:9000 --repo1-s3-key=<redacted> --repo1-s3-key-
secret=<redacted> --repo1-s3-region=us-east-1 --repo1-s3-uri-style=path -
repo1-storage-ca-file=/certs/ca.crt --repo1-storage-verify-tls --repo1-
type=s3 --stanza=backup --start-fast --stop-auto --type=full
2022-06-29 19:59:12.117 P00 INFO: execute non-exclusive pg_start_backup
(): backup begins after the requested immediate checkpoint completes
....
2022-06-29 20:17:58.091 P00 INFO: expire command end: completed
successfully (1073018ms)
```

2. OpenSearch manual backup.

Manual backup of Opensearch:

```
./backup-tool.sh platform backup-opensearch --help
# Usage: ./backup-tool.sh platform backup-opensearch [options]
#
# Options:
# --shared-namespace      Shared namespace
# --backup-bucket          Bucket with backups
# --admin-username         Opensearch admin username
# --admin-password         Opensearch admin password
# --help                  Help

# Example
./backup-tool.sh platform backup-opensearch --shared-namespace arturva-
shared-master-20220117-77 --admin-username admin --admin-password admin --
backup-bucket oss-archive
Creating of backup: 2022_06_29_22_12_16 started
Snapshot in progress
Snapshot executed successfully
```

3. GitLab manual backup script.

Manual backup of GitLab

```
./backup-tool.sh platform backup-gitlab --help
# Usage: ./backup-tool.sh platform backup-gitlab [options]
#
# Options:
# --shared-namespace      New shared namespace
# --help                  Help
```

```
# Example
./backup-tool.sh platform backup-gitlab --shared-namespace arturva-shared-master-20220117-77

job.batch/gitlab-task-runner-backup-manual created
Job in progress current state: Pending
Job in progress current state: Running
Begin parsing .erb templates from /var/opt/gitlab/templates
...
Deleting old backup s3://gitlab-backups-storage/1642842011_2022_01_22_14.3.2_gitlab_backup.tar
[DONE] Finished pruning old backups
Job in progress current state: Succeeded
job.batch "gitlab-task-runner-backup-manual" deleted
Job finished with status: Succeeded
```

4. Backup All (PostgreSQL+Opensearch+GitLab)

Manual backup ALL

```
./backup-tool.sh platform backup-all --help
```

Usage: ./backup-tool.sh platform backup-opensearch [options]

* - required

Options:

--shared-namespace	[*]	Shared namespace on which perform components backup
--oss-backup-bucket	[]	Bucket with backups. Default oss-archive.
--oss-admin-username	[*]	Opensearch admin username
--oss-admin-password	[*]	Opensearch admin password
--pg-backup-type	[]	Type of backup to do. Allowed values: ["full", "diff", "incr"]. Default full.
--help		Help

Example

```
./backup-tool.sh platform backup-all --shared-namespace {ns you want to backup} --oss-admin-username admin --oss-admin-password admin
```

=== Postgres backup started ===

Backuping started.

```
2024-02-21 15:35:35.456 P00 INFO: backup command begin 2.39: --delta --exclude=postgresql.auto.conf --exec-id=89845-b792af3e --io-timeout=900 --log-level-console=info --log-level-file=debug --log-path=/tmp --pg1-path=/stolon-data/postgres --process-max=2 --repo1-bundle --repo1-bundle-limit=10MiB --repo1-bundle-size=30MiB --repo1-path=/primary --repo1-retention-archive=7 --repo1-retention-archive-type=incr --repo1-retention-full=1 --repo1-retention-history=0 --repo1-s3-bucket=pg-archive --repo1-s3-endpoint=minio:9000 --repo1-s3-key=<redacted> --repo1-s3-key-secret=<redacted> --repo1-s3-
```

```

region=us-east-1 --repo1-s3-uri-style=path --repo1-storage-ca-
file=/certs/ca.crt --repo1-storage-verify-tls --repo1-type=s3 --stanza=backup
--start-fast --stop-auto --type=full
...
Backup finished.

=== Postgres backup finished ===

=== Opensearch backup started ===
...
Snapshot executed successfully

=== Opensearch backup finished ===

=== Gitlab backup started ===
...
[DONE] Finished pruning old backups
Job in progress current state: Succeeded
job.batch "gitlab-toolbox-backup-manual" deleted
Job finished with status: Succeeded

=== Gitlab backup finished ===

```

10.2.10.2. External Backups

1. MinIO manual backup to Google Storage.

Required to install Rclone to do manual backups to Google Storage. Installation guide - <https://rclone.org/install/>

You can use a backup script in different ways:

- a. Configure Rclone connection with MinIO and GCP on your own. And provide connection aliases to use:
 - i. MinIO configuration guide - <https://rclone.org/s3>.
 - ii. Google cloud storage configuration guide - <https://rclone.org/googlecloudstorage/>
- b. Provide required credentials to script and it will auto-generate the configuration for a single execution.

Auto generation of Rclone config to google storage is possible in three specified ways:

- a. With credentials as a file. In this case, you should specify a path to file with credentials to the service account that is granted the required permissions (read/write on needed bucket).
- b. With an iam-account if you are logged in as iam account user in your environment.
- c. With your own user credentials, if you logged in to your gcloud account.

Manual minio backup to gcp

```
./backup-tool.sh external google-storage backup --help

# Options:
# --rclone-minio-alias          Provide rclone minio alias if
rclone is configured or provide credentials directly.
# --rclone-gcp-alias           Provide rclone google cloud storage
alias if rclone is configured or provide credentials directly.
# --gcp-auth-type              Define auth type if not gcp rclone
alias provided ["file", "iam-account", "user"]
# --gcp-credentials-file       Provide path to credentials file if
--gcp-auth-type=file
# --gcp-bucket                 Google cloud storage bucket
# --minio-host                 Minio host if rclone alias for
minio not provided
# --minio-admin-access-key     Source minio admin access key if
rclone alias for minio not provided
# --minio-admin-secret-key     Source minio admin secret key if
rclone alias for minio not provided
# --shared-namespace           Shared namespace
# --help                       Help

# Example with configured rclone.
./backup-tool.sh external google-storage backup --rclone-gcp-alias gcp --
rclone-minio-alias minio --shared-namespace arturva-shared-master-20220117-77
--gcp-bucket minio-backups-arturva-shared-master-20220117-77

=== Backup bucket: gitlab-artifacts-storage started ===
...
=== Backup bucket: thetaray-public-artur-test1 finished ===

Sending log to logstash
ok
Completed

# Example with auto generated configuration by user credentials.
./backup-tool.sh external google-storage backup --gcp-auth-type user --minio-
host https://minio-arturva-shared-master-20220117-77.ndp.thetaraydev.com --
minio-admin-access-key thetaray --minio-admin-secret-key thetaray --shared-
namespace arturva-shared-master-20220117-77 --gcp-bucket minio-backups-
```

```
arturva-shared-master-20220117-77

=== Backuping bucket: gitlab-artifacts-storage started ===
...
=== Backuping bucket: thetaray-public-artur-test1 finished ===

Sending log to logstash
ok
```

2. MinIO manual backup - Azure Blob Storage.

Required to install Rclone to do manual backups to Google Storage. Installation guide - <https://rclone.org/install/>

You can use a backup script in different ways:

- a. Configure Rclone connection with MinIO and GCP on your own. And provide connection aliases to use.
 - i. Minio configuration guide - <https://rclone.org/s3/>.
 - ii. Google cloud storage configuration guide - <https://rclone.org/googlecloudstorage/>
- b. Provide required credentials to script and it will auto-generate the configuration for a single execution.

Auto-generating of Rclone config to blob storage is possible in two ways that you can specify.

- i. With account credentials --account-key and --account-name
- ii. With shared access token --sas-url and --account-name.

Manual minio backup to Azure:

```
./backup-tool.sh external azure-blob backup --help

# Options:
# --rclone-minio-alias          Provide rclone minio alias if
rclone is configured or provide credentials directly.
# --rclone-azure-alias          Provide rclone azure storage alias
if rclone is configured or provide credentials directly.
# --azure-auth-type              Define auth type if not azure
rclone alias provided ["credentials", "sas-url"]
# --sas-url                      Shared access signature for account
that have access to the storage for --azure-auth-type=sas-url
# --account-name                 Account name that have access to
the storage for --azure-auth-type=credentials
# --account-key                  Key for account that have access to
the storage for --azure-auth-type=credentials
```



```
# --sp_tenant_id           Optional - service principal tenant
id
# --sp_client_id          Optional - service principal client
id
# --sp_client_secret      Optional - service principal client
secret
# --azure-bucket          Azure storage bucket
# --minio-host            Minio host if rclone alias for
minio not provided
# --minio-admin-access-key Source minio admin access key if
rclone alias for minio not provided
# --minio-admin-secret-key Source minio admin secret key if
rclone alias for minio not provided
# --shared-namespace      Shared namespace
# --help                  Help

# Example with configured rclone.
./backup-tool.sh external azure-blob backup --rclone-azure-alias azure --
rclone-minio-alias minio --shared-namespace arturva-shared-master-20220208-
1171 --azure-bucket minio-backups-arturva-shared-master-20220208-1171

=== Backup bucket: gitlab-artifacts-storage started ===
...
=== Backup bucket: pg-archive finished ===

Sending log to logstash
ok
Completed

# Example with auto generated configuration with sas-url.
./backup-tool.sh external azure-blob backup --azure-auth-type sas-url --
minio-host https://minio-arturva-shared-master-20220208-
1171.ndp.thetaraydev.com --minio-admin-access-key thetaray --minio-admin-
secret-key thetaray --azure-bucket minio-backups-arturva-shared-master-
20220208-1171 --sas-url "https://drtestsa.blob.core.windows.net?{token}" --
shared-namespace arturva-shared-master-20220208-1171
=== Backup bucket: gitlab-artifacts-storage started ===
...
=== Backup bucket: pg-archive finished ===

Sending log to logstash
ok
Completed
```

3. Backup MinIO backups to S3.

Required to install Rclone to do manual backups to Google Storage. Installation guide - <https://rclone.org/install/>

You can use a backup script in different ways.

- Configure Rclone connection with MinIO and your S3 storage on your own. And provide connection aliases to use. S3 configuration guide - <https://rclone.org/s3>.
- Provide required credentials to script and it will auto-generate the configuration for a single execution.

Backup to Azure BLOB

```
./backup-tool.sh external s3 backup --help

# Options:
#   --rclone-minio-alias          Provide rclone minio alias if
rclone is configured or provide credentials directly.
#   --rclone-s3-alias            Provide rclone s3 storage alias if
rclone is configured or provide credentials directly.
#   --s3-bucket                  S3 storage bucket if rclone alias
for s3 not provided
#   --s3-endpoint                S3 storage endpoint if rclone alias
for s3 not provided
#   --s3-access-key-id          S3 storage admin access key if
rclone alias for s3 not provided
#   --s3-secret-access-key      S3 storage admin access key if
rclone alias for s3 not provided
#   --s3-provider                S3 storage provider (Minio, AWS
etc) if rclone alias for s3 not provided
#   --s3-additional-config      JSON S3 additional configuration
required by rclone depend on provider. Check documentation.
#   --minio-host                Minio host if rclone alias for
minio not provided
#   --minio-admin-access-key    Source minio admin access key if
rclone alias for minio not provided
#   --minio-admin-secret-key    Source minio admin secret key if
rclone alias for minio not provided
#   --shared-namespace          Shared namespace
#   --sp_tenant_id              Optional - service principal tenant
id
#   --sp_client_id              Optional - service principal client
id
#   --sp_client_secret          Optional - service principal client
secret
#   --help                      Help

# Example with configured rclone.
./backup-tool.sh external s3 backup --rclone-minio-alias minio1 --rclone-s3-
alias s3 --s3-bucket minio-backups-arturva-shared-master-20220328-1698
=== Backuping bucket: gitlab-artifacts-storage started ===
...
=== Backuping bucket: pg-archive finished ===
```

```

Sending log to logstash
ok
Completed

# Example with auto generated configuration with sas-url.
./backup-tool.sh external s3 backup --s3-bucket gitlab-uploads-storage --s3-
endpoint minio-arturva-shared-master-1654097857-2925.ndp.thetaraydev.com --
s3-access-key-id thetaray --s3-secret-access-key thetaray --s3-provider Minio
--minio-host minio-arturva-shared-master-20220328-1698.ndp.thetaraydev.com --
minio-admin-access-key thetaray --minio-admin-secret-key thetaray --shared-
namespace arturva-shared-master-20220328-1698
=== Backuping bucket: gitlab-artifacts-storage started ===
...
=== Backuping bucket: pg-archive finished ===

Sending log to logstash
ok
Completed

```

10.2.11. Restore from Backup

Restoring an environment from backup is made over a few steps:

Step 1. Create a new namespace on the cluster where you want to restore to, from backup.

Step 2. Using the same configuration as was setup on a previous namespace, deploy only minio release to your environment by running the following script:

```

./install-shared-replication.sh --new-shared-namespace arturva-shared-master-
20220117-78 --customer customer

```

After the helm release is installed, verify that in your replication environment, minio is deployed:

Setup minio

```

kubectl -n { ns } get po

```

NAME	READY	STATUS	RESTARTS
AGE			
datastores-bootstrap-replication-conf-prjv6 0/1	Completed	0	105s
minio-0	1/1	Running	0 3m17s
minio-1	1/1	Running	0 3m17s

```
minio-
2                               1/1    Running    0          3m17s
minio-
3                               1/1    Running    0          3m17s
```

Step 3. Restore minio from external storage.

Google Storage

You can use a backup script in different ways.

1. Configure Rclone connection with MinIO and GCP on your own and provide connection aliases to use.
 - a. Minio configuration guide - <https://rclone.org/s3>.
 - b. Google cloud storage configuration guide - <https://rclone.org/googlecloudstorage/>
2. Provide required credentials to script and it will auto-generate the configuration for a single execution.

Auto generating of Rclone config to google storage is possible in three ways specified ways:

1. With credentials as a file. In this case, you should specify a path to file with credentials to the service account that have proper permissions (read/write on needed bucket).
2. With iam-account if you are logged in as iam account in your environment.
3. And with your own user credentials if you logged in to your gcloud account.

Manual minio backup to gcp

```
./backup-tool.sh external google-storage restore --help

# Options:
# --rclone-minio-alias          Provide rclone minio alias if
rclone is configured or provide credentials directly.
# --rclone-gcp-alias           Provide rclone google cloud storage
alias if rclone is configured or provide credentials directly.
# --gcp-auth-type              Define auth type if not gcp rclone
alias provided ["file", "iam-account", "user"]
# --gcp-bucket                 Google cloud storage bucket
# --gcp-credentials-file       Provide path to credentials file if
--gcp-auth-type=file
# --minio-host                 Minio host if rclone alias for
minio not provided
# --minio-admin-access-key      Source minio admin access key if
rclone alias for minio not provided
# --minio-admin-secret-key      Source minio admin secret key if
```

```
rclone alias for minio not provided
# --retry-count           Retries to do if bucket sync
failed. 3 by default.
# --help                 Help

# Example with configured rclone.
./backup-tool.sh external google-storage restore --rclone-gcp-alias gcp --
rclone-minio-alias minio --shared-namespaces arturva-shared-master-20220117-77
--gcp-bucket minio-backups-arturva-shared-master-20220117-77
```

Azure Blob Storage

Required to install rclone to do manual backups to Azure Blob Storage. Installation guide - <https://rclone.org/install/>

You can use a backup script in different ways:

1. Configure Rclone connection with MinIO and GCP on your own. And provide connection aliases to use.
 - a. Minio configuration guide - <https://rclone.org/s3>.
 - b. Google cloud storage configuration guide - <https://rclone.org/googlecloudstorage/>
2. Provide required credentials to script and it will auto-generate the configuration for a single execution.

Auto-generating of Rclone config to blob storage is possible in two specified ways:

1. With account credentials --account-key and --account-name.
2. With shared access token --sas-url and --account-name.

Restore from Azure BLOB

```
./backup-tool.sh external azure-blob restore --help

# Options:
# --rclone-minio-alias      Provide rclone minio alias if
rclone is configured or provide credentials directly.
# --rclone-azure-alias      Provide rclone azure storage alias
if rclone is configured or provide credentials directly.
# --azure-auth-type          Define auth type if not azure
rclone alias provided ["credentials", "sas-url"]
# --sas-url                  Shared access signature for account
that have access to the storage for --azure-auth-type=sas-url
# --account-name             Account name that have access to
the storage for --azure-auth-type=credentials
# --account-key              Key for account that have access to
the storage for --azure-auth-type=credentials
# --sp_tenant_id            Optional - service principal tenant
```

```

id
# --sp_client_id           Optional - service principal client
id
# --sp_client secret.      Optional - service principal client
secret
# --azure-bucket           Azure storage bucket
# --minio-host             Minio host if rclone alias for
minio not provided
# --minio-admin-access-key Source minio admin access key if
rclone alias for minio not provided
# --minio-admin-secret-key Source minio admin secret key if
rclone alias for minio not provided
# --shared-namespace      Shared namespace
# --help                  Help

# Example with configured rclone.
./backup-tool.sh external azure-blob restore --rclone-azure-alias azure --
rclone-minio-alias minio --shared-namespace arturva-shared-master-20220208-
1171 --azure-bucket minio-backups-arturva-shared-master-20220208-1171

```

S3 Compatible Storage

You can use a backup script in different ways.

1. Configure Rclone connection with MinIO and your S3 storage on your own. And provide connection aliases to use. S3 configuration guide - <https://rclone.org/s3>.
2. Provide required credentials to script and it will auto-generate the configuration for a single execution.

Manual minio backup to Azure

```

./backup-tool.sh external s3 restore --help

# Options:
# --rclone-minio-alias      Provide rclone minio alias if
rclone is configured or provide credentials directly.
# --rclone-s3-alias         Provide rclone s3 storage alias if
rclone is configured or provide credentials directly.
# --s3-bucket               S3 storage bucket if rclone alias
for s3 not provided
# --s3-endpoint             S3 storage endpoint if rclone alias
for s3 not provided
# --s3-access-key-id        S3 storage admin access key if
rclone alias for s3 not provided
# --s3-secret-access-key    S3 storage admin access key if
rclone alias for s3 not provided

```

```
# --s3-provider                S3 storage provider (Minio, AWS
etc) if rclone alias for s3 not provided
# --s3-additional-config       JSON S3 additional configuration
required by rclone depend on provider. Check documentation.
# --minio-host                 Minio host if rclone alias for
minio not provided
# --minio-admin-access-key     Source minio admin access key if
rclone alias for minio not provided
# --minio-admin-secret-key     Source minio admin secret key if
rclone alias for minio not provided
# --shared-namespace           Shared namespace
# --help                       Help

# Example with configured rclone.
./backup-tool.sh external s3 restore --rclone-minio-alias minio1 --rclone-s3-
alias s3 --s3-bucket minio-backups-arturva-shared-master-20220328-1698
```

Step 4. Now when minio deployed on a replicated namespace has all required backups we can recover the environment.

To do this run shared master installation script with `--recover-cluster true` argument.

You can also specify either `--pg-pitr` or `--pg-backup-name` to restore from data in MINIO.

Note: However - if you have done point in time restore (PITR) from cloud provider then there is no need to also specify same `--pg-pitr` here since this environment's MINIO is at the respective point in time from the cloud already.

Important. Before installation verify that gitlab secrets is the same as the secrets of main gitlab. File `shared-certs.yaml.gotmpl`: key `gitlab_instance_certs`, file `shared-secrets.yaml.gotmpl`: key `"gitlab_instance_secrets"`

Recover Cluster

```
./install-shared.sh --new-shared-namespace arturva-shared-master-20220119-79
--recover-cluster true --customer customer

# Check that everything up
kubectl -n arturva-shared-master-20220119-79 get po
NAME                                READY   STATUS    RESTART
S   AGE
datastores-bootstrap-replication-conf-
prjv6                               0/1     Completed 0          39m
```

```

gitlab-bootstrap-gitlab-
ghzgc          0/1      Completed    0          2d17h
gitlab-gitaly-
0              1/1      Running      0          7m48s
gitlab-migrations-1-
tdfrr          0/1      Completed    4          7m48s
gitlab-sidekiq-all-in-1-v1-64488c74f4-
f8zs5          1/1      Running      0          7m48s
gitlab-task-runner-767b6db675-
zjn7b          1/1      Running      0          7m48s
gitlab-webservice-default-5674f58bbf-
zlblw          2/2      Running      0          7m48s
hasura-6c9b9699d5-
6xgk8          2/2      Running      6          8m9s
keycloak-
0              1/1      Running      4          8m8s
keycloak-bootstrap-keycloak-
nfhlj          0/1      Completed    0          8m6s
logstash-logstash-
0              1/1      Running      0          8m8s
minio-
0              1/1      Running      0          5m
43s
minio-
1              1/1      Running      0          6m
14s
minio-
2              1/1      Running      0          7m
23s
minio-
3              1/1      Running      0          7m
50s
opensearch-bootstrap-
xh46d          0/1      Completed    0          8m7s
opensearch-cluster-master-
0              1/1      Running      0          8m8s
opensearch-dashboards-76bdc6764d-
9thrv          1/1      Running      0          8m8s
pgadmin4-789fb6c598-
bgjvj          1/1      Running      0          7m56s
postgres-create-cluster-
qqqmc          0/1      Completed    0          4m25s
postgres-keeper-
0              1/1      Running      0          7m54s
postgres-proxy-5f9b4d6bf8-
wvbm8          1/1      Running      0          7m56s
postgres-sentinel-7695c7bb69-
b5vwk          1/1      Running      0          7m55s
redis-master-

```


0	1/1	Running	0	7m55s
---	-----	---------	---	-------

After installation run the restoration script. You can provide a backup name you want to use in another case it will use the latest backups.

Check the list of GitLab backups you can at MinIO bucket where they are stored..

The list of OpenSearch snapshots you can see at the monitoring service by requesting ndpoint: "_snapshot/{bucket}/_all" or execute this command:

```
kubectl -n {namespace} exec -it opensearch-cluster-master-0 bash -- curl -k https://{user}:{password}@localhost:9200/_snapshot/{bucket}/_all\?pretty\=true
```

Note: If you don't specify backup name it will automatically use the latest.

GitLab Restoration

```
./backup-tool.sh promote-failover-site --help
```

Options:

--shared-namespace	Shared namespace
--gitlab-backup	Gitlab backup to restore (_gitlab_backup.tar should be omitted from the name). Latest by default.
--minio-host	Minio host
--minio-admin-access-key	Source minio admin access key
--minio-admin-secret-key	Source minio admin secret key
--gitlab-bucket	Bucket with gitlab backups
--oss-backup	Opensearch backup to restore. Latest by default.
--oss-admin-username	Opensearch admin user.
--oss-admin-password	Opensearch admin password.
--oss-backup-bucket	Opensearch backup bucket.
--help	Help

```
./backup-tool.sh promote-failover-site --shared-namespace arturva-shared-master-1656515226-3627 --minio-host https://minio-arturva-shared-master-1656515226-3627.ndp.thetaraydev.com --minio-admin-access-key admin --minio-admin-secret-key admin --gitlab-bucket gitlab-backups-storage --oss-admin-username admin --oss-admin-password admin --oss-backup-bucket oss-archive
Recovering gitlab started
Added `source` successfully.
Latest backup selected 1643025604_2022_01_24_14.3.2
...
Recovering gitlab finished
```

```
Recovering opensearch started
Recovering of backup: 2022_01_24_13_00_44 started
{"acknowledged":true}{
  "accepted" : true
}
Recovering opensearch finished
```

Now you got a recovered and working shared environment.

Finally, you need to create new namespaces for platform, exec and application (for each solution), and then just run the regular installation flow using the restored shared namespace.

10.2.12. Point in time restore of external storage

Point in time restore of external storage means rollback bucket state to specific date. For example if something happened with current environment and retained backups are applicable to restore to the working state you have an ability to rollback external storage to specific date in range of configured versioning retention and use needed backups.

This feature is available for all types of external storage.

10.2.12.1. How can I know which point of time I want to restore? (logging)

Information is logged about each action (backup and pitr itself) that can be used as a restore point. Logs are stored to opensearch "events" index and logs bucket in cloud storage that is named logs-{external storage bucket name}.

You can use command line tool in case if you want to check if last backup was successful or understand the reason for failure in case there is something wrong OR you want to do a restore from backup in the case of a corrupted environment that is still running.

```
./backup-tool.sh external {storage type} info --help

# Options:
# --shared-namespace          Shared namespace
# --size                      Number of last backup
# --oss-admin-username        Opensearch admin user.
# --oss-admin-password        Opensearch admin password.
# --help                      Help

./backup-tool.sh external {storage} info --shared-namespace arturva-shared-
master-20220328-1698 --oss-admin-password admin --oss-admin-username admin --
size 2
```

```
[
  {
    "shared_master": "arturva-shared-master-20220328-1698",
    "source": "minio",
    "target": "s3_storage",
    "bucket": "minio-backups-artur-arturva-shared-master-20220328-1698",
    "event_type": "backup",
    "success": true,
    "manual": false,
    "timestamp": "2022-08-12T22:32:52Z"
  },
  {
    "shared_master": "arturva-shared-master-20220328-1698",
    "target": "s3_storage",
    "bucket": "minio-backups-artur-arturva-shared-master-20220328-1698",
    "event_type": "pitr",
    "success": true,
    "timestamp": "2022-08-12T22:52:50Z",
    "pitr_timestamp": "2022-08-12T22:38:14Z"
  }
]
```

In the code above you can see "timestamp" field represents a point in time you can refer to.

This is the recommended way to inspect backups. If a corrupted environment is down, you have another option to check why.

Go to your storage page and find bucket named logs-{backup bucket name}. This is a bucket created during deployment where JSON logs are stored to.

The figure below shows an example.

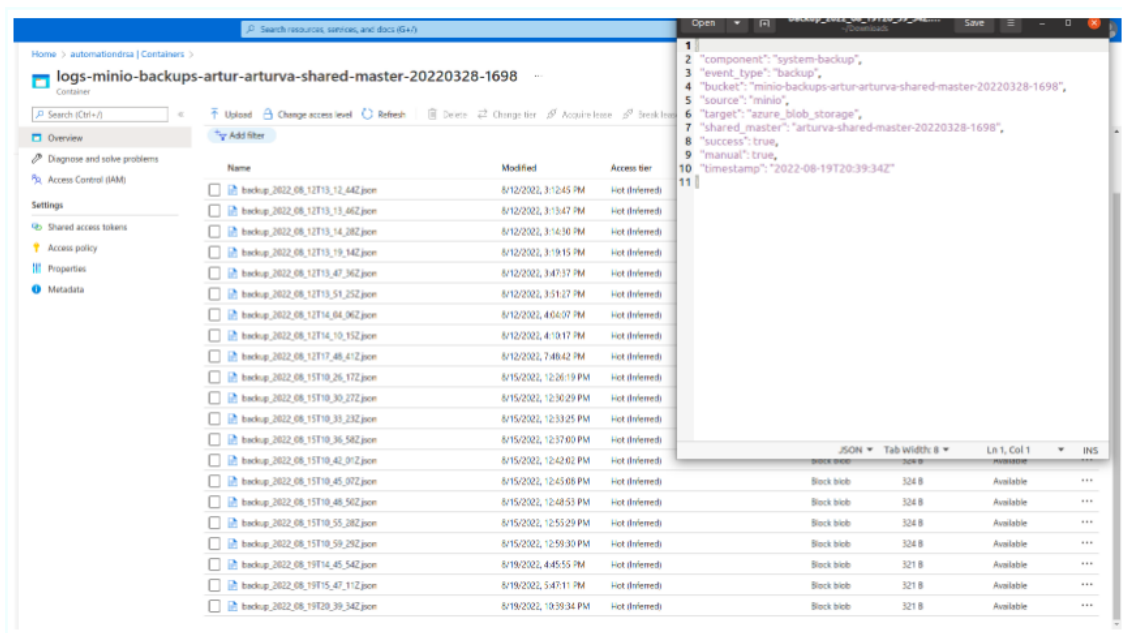


Figure 4: Each file represent log of action that creates point in time that gave ability to restore.

10.2.12.2. How can I perform PITR?

You are able to rollback your bucket state in days range configured during deployment. To do so ,check the appropriate configuration section of this documentation.

Google cloud storage

```
./backup-tool.sh external google-storage pitr

# Options:
# --gcp-auth-type          Define auth type if not gcp rclone
alias provided ["file", "iam-account"]
# --credentials-file       Provide path to credentials file if
--gcp-auth-type=file
# --bucket                Google cloud storage bucket
# --shared-namespace       Shared namespace
# --timestamp              Timestamp in format of output from
backup-tool external google-storage info cmd.
# --help                  Help
```

```
./backup-tool.sh external google-storage pitr --gcp-auth-type file --
credentials-file creds.json --bucket {backup bucket} --shared-namespace {ns
that is backedup} --timestamp 2022-08-12T22:52:50Z
```

Azure blob storage

```
./backup-tool.sh external azure-blob pitr

# Options:
#   --container                Container to perform pitr
#   --storage-account          Customer storage account
#   --resource-group            Customer resource group
#   --subscription             Azure subscription if
#   --shared-namespace         Shared namespace
#   --timestamp                Timestamp in format of
#   --sas-token                Shared access token.
#   --help                    Help

./backup-tool.sh external azure-blob pitr --container {backup container} --
storage-account {account name} --timestamp 2022-08-15T10:42:11Z --
subscription {subscription} --shared-namespace {ns that is backedup} --sas-
token "{token}"
```

S3 storage

```
./backup-tool.sh external s3 pitr

# Options:
#   --s3-endpoint              Provide s3 endpoint in format
#   --s3-admin-access-key      S3 admin access key.
#   --s3-admin-secret-key      S3 admin secret key.
#   --s3-provider              S3 storage provider (Minio, AWS
#   --s3-additional-config      JSON S3 additional configuration
#   --bucket                   Bucket to perform pitr on.
#   --timestamp                Timestamp in format of output from
#   --shared-namespace         Shared namespace
#   --help                    Help

./backup-tool.sh external s3 pitr --s3-endpoint {your s3 endpoint} --s3-
```

```
access-key-id {your s3 access key} --s3-secret-access-key {your s3 secret  
key} --s3-provider Minio --bucket {backup bucket} --timestamp 2022-08-  
19T15:15:16Z --shared-namespace {ns that is backedup}
```

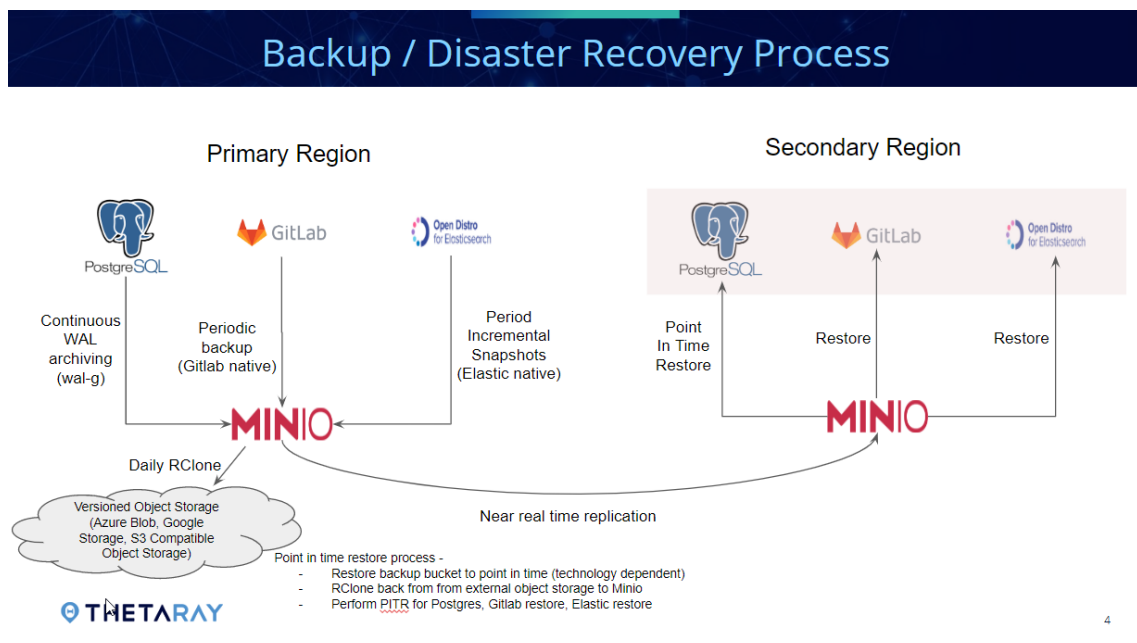
10.2.13. Disaster Recovery

As detailed in the Overview, Disaster Recovery is a mechanism that enables ongoing replication of data managed by the system to an secondary data center, allowing the system to be recovered in case of a disaster within the primary data center.

Replication should be configured between 2 Minio instances deployed on clusters located at different regions (Primary and Secondary region). Replication enables with specific script manually (technical details provided below).

Replication means that when we upload or delete data to primary Minio the same action is completed on the secondary Minio.

The following image provides an illustration how backups and replication works.



10.2.14. Setup Environment

Required to install rclone. Installation guide - <https://rclone.org/install/>

To perform disaster recovery you should to setup additional replication environment. with a different cluster from the main environment cluster.

Step 1. Deploy only minio release to your replication environment. To complete this step , run this script.

```
./install-shared-replication.sh --new-shared-namespace <infra namespace name>
--env-profile <env profile> --cluster-domain <cluster domain> [--create-
namespace false] --customer <customer-bucket-name>
```

After helm release is installed, you can check now that in your replication environment, minio is deployed.

Setup minio

```
kubectl -n { ns } get po
```

NAME	READY	STATUS	RESTARTS	AGE
datastores-create-replication-user-zqmr1 0/1	Completed	0	105s	
minio-0	1/1	Running	0	3m17s
minio-1	1/1	Running	0	3m17s
minio-2	1/1	Running	0	3m17s
minio-3	1/1	Running	0	3m17s

Step 2. Configure bucket replication between two environments main and replication. The script to complete this task is located in the backup-recovery folder.

Setup mini replication

```
./backup-tool.sh replication setup --help
```

Options:

--source-host	Source minio cluster host
--destination-host	Destination minio cluster host
--source-minio-admin-access-key	Source minio admin access key
--source-minio-admin-secret-key	Source minio admin secret key
--destination-minio-admin-access-key	Source minio admin access key
--destination-minio-admin-secret-key	Source minio admin secret key
--minio-replication-user-access-key	Minio replication access key
--minio-replication-user-secret-key	Minio replication access key
--progress	Show progress of bucket replicating
--help	Help

Example

```
./backup-tool.sh replication setup --source-host minio-arturva-shared-master-20220328-1698.ndp.thetaraydev.com --destination-host minio-arturva-shared-master-20220117-78.ndp.thetaraydev.com --source-minio-admin-access-key thetaray --source-minio-admin-secret-key thetaray --destination-minio-admin-access-key thetaray --destination-minio-admin-secret-key thetaray --minio-replication-user-access-key replication --minio-replication-user-secret-key replication
```

```
=== Enabling replication for bucket: gitlab-artifacts-storage/ started ===
gitlab-artifacts-storage
```



```
...
=== Enabling replication for bucket: thetaray-public-artur-test1/ finished
===
```

Important: Be aware, that bucket replication is enabled only on existing buckets. This means, if you installed a new solution on your environment and run this script earlier, you are required to run it again in order to configure the replication on newly created solution buckets.

10.2.15. Recovering Environment

When minio is deployed with the replicated namespace, with all required backups, the environment can be recovered.

To do this run shared master installation script with `--recover-cluster true` argument.

Note: Important. Before installation be sure that gitlab secrets is the same with the secrets of main gitlab. File `shared-certs.yaml.gotmpl`: key `gitlab_instance_certs`, file `shared-secrets.yaml.gotmpl`: key `"gitlab_instance_secrets"`.

Recover cluster:

```
./install-shared.sh --new-shared-namespace arturva-shared-master-20220119-79
--recover-cluster true --customer customer

# Check that everything up
kubectl -n arturva-shared-master-20220119-79 get po
NAME                                READY   STATUS    RESTART
S   AGE
datastores-bootstrap-replication-conf-
prjv6                                0/1     Completed 0          39m
gitlab-bootstrap-gitlab-
ghzgc                                0/1     Completed 0          2d17h
gitlab-gitaly-
0                                     1/1     Running   0          7m48s
gitlab-migrations-1-
tdfrr                                0/1     Completed 4          7m48s
gitlab-sidekiq-all-in-1-v1-64488c74f4-
f8zs5                                1/1     Running   0          7m48s
gitlab-task-runner-767b6db675-
zjn7b                                1/1     Running   0          7m48s
gitlab-webservice-default-5674f58bbf-
zlblw                                2/2     Running   0          7m48s
```

```

hasura-6c9b9699d5-
6xgk8                2/2    Running    6          8m9s
keycloak-
0                    1/1    Running    4          8m8s
keycloak-bootstrap-keycloak-
nfhlj                0/1    Completed  0          8m6s
logstash-logstash-
0                    1/1    Running    0          8m8s
minio-
0                    1/1    Running    0          5m
43s
minio-
1                    1/1    Running    0          6m
14s
minio-
2                    1/1    Running    0          7m
23s
minio-
3                    1/1    Running    0          7m
50s
opensearch-bootstrap-
xh46d                0/1    Completed  0          8m7s
opensearch-cluster-master-
0                    1/1    Running    0          8m8s
opensearch-dashboards-76bdc6764d-
9thrv                1/1    Running    0          8m8s
pgadmin4-789fb6c598-
bgjvj                1/1    Running    0          7m56s
postgres-create-cluster-
qqqmc                0/1    Completed  0          4m25s
postgres-keeper-
0                    1/1    Running    0          7m54s
postgres-proxy-5f9b4d6bf8-
wvbm8                1/1    Running    0          7m56s
postgres-sentinel-7695c7bb69-
b5vwk                1/1    Running    0          7m55s
redis-master-
0                    1/1    Running    0          7m55s

```

After running the installation restoration script, provide a backup name which you require can be used e at another case. It will automatically use the latest backups. The list of gitLab backups can be viewed and verified from the minio bucket, where they are stored.

The list of Opensearch snapshots can be viewed at the monitoring service, by making a request to the endpoint: "_snapshot/{bucket}/_all" or executing the followig command:

```
kubectl -n {namespace} exec -it opensearch-cluster-master-0 bash -- curl -k
https://{user}:{password}@localhost:9200/_snapshot/{bucket}/_
all\?pretty\=true
```

Note: If you don't specify a backup name, it will automatically use the latest.

GitLab Restoration

```
./backup-tool.sh promote-failover-site --help
```

Options:

--shared-namespace	Shared namespace
--gitlab-backup	Gitlab backup to restore (_gitlab_
backup.tar should be omitted from the name). Latest by default.	
--minio-host	Minio host
--minio-admin-access-key	Source minio admin access key
--minio-admin-secret-key	Source minio admin secret key
--gitlab-bucket	Bucket with gitlab backups
--oss-backup	Opensearch backup to restore. Latest
by default.	
--oss-admin-username	Opensearch admin user.
--oss-admin-password	Opensearch admin password.
--oss-backup-bucket	Opensearch backup bucket.
--help	Help

```
./backup-tool.sh promote-failover-site --shared-namespace arturva-shared-
master-1656515226-3627 --minio-host https://minio-arturva-shared-master-
1656515226-3627.ndp.thetaraydev.com --minio-admin-access-key admin --minio-
admin-secret-key admin --gitlab-bucket gitlab-backups-storage --oss-admin-
username admin --oss-admin-password admin --oss-backup-bucket oss-archive
Recovering gitlab started
Added `source` successfully.
Latest backup selected 1643025604_2022_01_24_14.3.2
...
Recovering gitlab finished
Recovering opensearch started
Recovering of backup: 2022_01_24_13_00_44 started
{"acknowledged":true}{
  "accepted" : true
}
Recovering opensearch finished
```

That's it, you should now have a recovered and working cluster.

Finally, you need to create new namespaces for platform, exec and application (for each solution), and then just run the regular installation flow using the restored shared namespace.

10.2.16. Disaster Recover (DR) Summary

In summary, the basic DR setup flow is as follows:

1. Configure schedule in environment/{profile}/backups.yaml.gotmpl
2. Create replication namespace.
3. Install the replication environment that includes only MinIO release on created namespace with `./install-shared-replication.sh` script.
4. Enable replication between buckets with script `./setup-minio-replication.sh` and rerun on each newly created solution (new buckets).

The basic DR recovery flow is as follows:

1. Recover shared environment with `./install-shared` script but with flag: `--recover-cluster true`.
2. Run `promote-failover-site.sh`
3. Create platform, exec and, app namespaces and install it.

11. Appendix D - Prerequisites - OCP Specific

Important Note: In case Thetaray System is already installed with version prior to 6.10.2 (included) please remove all old rolebindings (V1)

For upgrade prior to 6.11 delete the rolebinding

```
kubectl delete rolebindings --all -n {shared-namespace}
kubectl delete rolebindings --all -n {platform-namespace}
kubectl delete rolebindings --all -n {exec-namespace}
kubectl delete rolebindings logstash-scc logstash-scc-v2 --ignore-not-found=true -n {apps-namespace}
```

TTR rolebinding configuration example

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: thetaray-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: thetaray-admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: thetaray-admin
```

The required permissions can be seen in yaml file located under:

```
environments/ocp-prod/thetaray-user-rbac.yaml
```

Note: "thetaray-admin" cluster role is based on the default "admin" cluster role with additional two sections of permissions for "securitycontextconstraints" "pods/finalizers" in the end

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: thetaray-admin
rules:
```

```

- apiGroups:
  - operators.coreos.com
  resources:
  - subscriptions
  verbs:
  - create
  - update
  - patch
  - delete
- apiGroups:
  - operators.coreos.com
  resources:
  - clusterserviceversions
  - catalogsources
  - installplans
  - subscriptions
  verbs:
  - delete
- apiGroups:
  - operators.coreos.com
  resources:
  - clusterserviceversions
  - catalogsources
  - installplans
  - subscriptions
  - operatorgroups
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - packages.operators.coreos.com
  resources:
  - packagemanifests
  - packagemanifests/icon
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - cert-manager.io
  resources:
  - certificates
  - certificaterequests
  - issuers
  verbs:
  - create

```

```

- delete
- deletecollection
- patch
- update
- apiGroups:
  - acme.cert-manager.io
resources:
- challenges
- orders
verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
  - cert-manager.io
resources:
- certificates
- certificaterequests
- issuers
verbs:
- get
- list
- watch
- apiGroups:
  - acme.cert-manager.io
resources:
- challenges
- orders
verbs:
- get
- list
- watch
- apiGroups:
  - logging.openshift.io
resources:
- clusterlogforwarders
verbs:
- '*'
- apiGroups:
  - logging.openshift.io
resources:
- clusterloggings
verbs:
- '*'
- apiGroups:

```

```

- logging.openshift.io
resources:
- clusterlogforwarders
verbs:
- create
- update
- patch
- delete
- apiGroups:
- logging.openshift.io
resources:
- clusterloggings
verbs:
- create
- update
- patch
- delete
- apiGroups:
- logging.openshift.io
resources:
- elasticsearches
verbs:
- create
- update
- patch
- delete
- apiGroups:
- logging.openshift.io
resources:
- kibanas
verbs:
- create
- update
- patch
- delete
- apiGroups:
- packages.operators.coreos.com
resources:
- packagemanifests
verbs:
- create
- update
- patch
- delete
- apiGroups:
- ""
resources:

```



```

- secrets
- serviceaccounts
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - ""
  - image.openshift.io
resources:
- imagestreamimages
- imagestreammappings
- imagestreams
- imagestreams/secrets
- imagestreamtags
- imagetags
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - ""
  - image.openshift.io
resources:
- imagestreamimports
verbs:
- create
- apiGroups:
  - ""
  - image.openshift.io
resources:
- imagestreams/layers
verbs:
- get
- update
- apiGroups:
  - ""

```

```

resources:
  - namespaces
verbs:
  - get
- apiGroups:
  - ""
  - project.openshift.io
resources:
  - projects
verbs:
  - get
- apiGroups:
  - ""
resources:
  - pods/attach
  - pods/exec
  - pods/portforward
  - pods/proxy
  - secrets
  - services/proxy
verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
resources:
  - serviceaccounts
verbs:
  - impersonate
- apiGroups:
  - ""
resources:
  - pods
  - pods/attach
  - pods/exec
  - pods/portforward
  - pods/proxy
verbs:
  - create
  - delete
  - deletecollection
  - patch
  - update
- apiGroups:
  - ""
resources:

```

```
- configmaps
- endpoints
- persistentvolumeclaims
- replicationcontrollers
- replicationcontrollers/scale
- secrets
- serviceaccounts
- services
- services/proxy
verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
  - apps
resources:
- daemonsets
- deployments
- deployments/rollback
- deployments/scale
- replicaset
- replicaset/scale
- statefulsets
- statefulsets/scale
verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
  - autoscaling
resources:
- horizontalpodautoscalers
verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
  - batch
resources:
- cronjobs
- jobs
```

```

verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
- extensions
resources:
- daemonsets
- deployments
- deployments/rollback
- deployments/scale
- ingresses
- networkpolicies
- replicaset
- replicaset/scale
- replicationcontrollers/scale
verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
- policy
resources:
- poddisruptionbudgets
verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
- networking.k8s.io
resources:
- ingresses
- networkpolicies
verbs:
- create
- delete
- deletecollection
- patch
- update
- apiGroups:
- metrics.k8s.io

```

```

resources:
  - pods
  - nodes
verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  - image.openshift.io
resources:
  - imagestreams
verbs:
  - create
- apiGroups:
  - ""
  - build.openshift.io
resources:
  - builds/details
verbs:
  - update
- apiGroups:
  - ""
  - build.openshift.io
resources:
  - builds
verbs:
  - get
- apiGroups:
  - snapshot.storage.k8s.io
resources:
  - volumesnapshots
verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
  - deletecollection
- apiGroups:
  - ""
  - build.openshift.io
resources:
  - buildconfigs
  - buildconfigs/webhooks

```

```

- builds
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - ""
  - build.openshift.io
resources:
- builds/log
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - build.openshift.io
resources:
- buildconfigs/instantiate
- buildconfigs/instantiatebinary
- builds/clone
verbs:
- create
- apiGroups:
  - build.openshift.io
resources:
- jenkins
verbs:
- edit
- view
- apiGroups:
  - ""
  - apps.openshift.io
resources:
- deploymentconfigs
- deploymentconfigs/scale
verbs:
- create
- delete
- deletecollection
- get
- list

```

```

- patch
- update
- watch
- apiGroups:
  - ""
  - apps.openshift.io
resources:
- deploymentconfigrollbacks
- deploymentconfigs/instantiate
- deploymentconfigs/rollback
verbs:
- create
- apiGroups:
  - ""
  - apps.openshift.io
resources:
- deploymentconfigs/log
- deploymentconfigs/status
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - image.openshift.io
resources:
- imagestreams/status
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - quota.openshift.io
resources:
- appliedclusterresourcequotas
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - route.openshift.io
resources:
- routes
verbs:
- create

```

```

- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - ""
  - route.openshift.io
resources:
- routes/custom-host
verbs:
- create
- apiGroups:
  - ""
  - route.openshift.io
resources:
- routes/status
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - template.openshift.io
resources:
- processedtemplates
- templateconfigs
- templateinstances
- templates
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - extensions
  - networking.k8s.io
resources:
- networkpolicies
verbs:
- create

```



```

- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - ""
  - build.openshift.io
resources:
- buildlogs
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - ""
resources:
- resourcequotausages
verbs:
- get
- list
- watch
- apiGroups:
  - apiextensions.k8s.io
resourceNames:
- clusterlogforwarders.logging.openshift.io
resources:
- customresourcedefinitions
verbs:
- get
- apiGroups:
  - logging.openshift.io
resources:
- clusterlogforwarders
verbs:
- get
- list
- watch
- apiGroups:
  - apiextensions.k8s.io

```

```
resourceNames:
- clusterloggings.logging.openshift.io
resources:
- customresourcedefinitions
verbs:
- get
- apiGroups:
- logging.openshift.io
resources:
- clusterloggings
verbs:
- get
- list
- watch
- apiGroups:
- apiextensions.k8s.io
resourceNames:
- elasticsearches.logging.openshift.io
resources:
- customresourcedefinitions
verbs:
- get
- apiGroups:
- logging.openshift.io
resources:
- elasticsearches
verbs:
- get
- list
- watch
- apiGroups:
- apiextensions.k8s.io
resourceNames:
- kibanas.logging.openshift.io
resources:
- customresourcedefinitions
verbs:
- get
- apiGroups:
- logging.openshift.io
resources:
- kibanas
verbs:
- get
- list
- watch
- apiGroups:
```

```

- packages.operators.coreos.com
resources:
- packagemanifests
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - image.openshift.io
resources:
- imagestreamimages
- imagestreammappings
- imagestreams
- imagestreamtags
- imagetags
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - image.openshift.io
resources:
- imagestreams/layers
verbs:
- get
- apiGroups:
  - ""
resources:
- configmaps
- endpoints
- persistentvolumeclaims
- persistentvolumeclaims/status
- pods
- replicationcontrollers
- replicationcontrollers/scale
- serviceaccounts
- services
- services/status
verbs:
- get
- list
- watch
- apiGroups:
  - ""
resources:

```

```

- bindings
- events
- limitranges
- namespaces/status
- pods/log
- pods/status
- replicationcontrollers/status
- resourcequotas
- resourcequotas/status
verbs:
- get
- list
- watch
- apiGroups:
  - ""
resources:
- namespaces
verbs:
- get
- list
- watch
- apiGroups:
  - apps
resources:
- controllerrevisions
- daemonsets
- daemonsets/status
- deployments
- deployments/scale
- deployments/status
- replicaset
- replicaset/scale
- replicaset/status
- statefulsets
- statefulsets/scale
- statefulsets/status
verbs:
- get
- list
- watch
- apiGroups:
  - autoscaling
resources:
- horizontalpodautoscalers
- horizontalpodautoscalers/status
verbs:
- get

```

```

- list
- watch
- apiGroups:
  - batch
resources:
- cronjobs
- cronjobs/status
- jobs
- jobs/status
verbs:
- get
- list
- watch
- apiGroups:
  - extensions
resources:
- daemonsets
- daemonsets/status
- deployments
- deployments/scale
- deployments/status
- ingresses
- ingresses/status
- networkpolicies
- replicaset
- replicaset/scale
- replicaset/status
- replicationcontrollers/scale
verbs:
- get
- list
- watch
- apiGroups:
  - policy
resources:
- poddisruptionbudgets
- poddisruptionbudgets/status
verbs:
- get
- list
- watch
- apiGroups:
  - networking.k8s.io
resources:
- ingresses
- ingresses/status
- networkpolicies

```

```

verbs:
- get
- list
- watch
- apiGroups:
- snapshot.storage.k8s.io
resources:
- volumesnapshots
verbs:
- get
- list
- watch
- apiGroups:
- ""
- build.openshift.io
resources:
- buildconfigs
- buildconfigs/webhooks
- builds
verbs:
- get
- list
- watch
- apiGroups:
- build.openshift.io
resources:
- jenkins
verbs:
- view
- apiGroups:
- ""
- apps.openshift.io
resources:
- deploymentconfigs
- deploymentconfigs/scale
verbs:
- get
- list
- watch
- apiGroups:
- ""
- route.openshift.io
resources:
- routes
verbs:
- get
- list

```

```

- watch
- apiGroups:
  - ""
  - template.openshift.io
resources:
- processedtemplates
- templateconfigs
- templateinstances
- templates
verbs:
- get
- list
- watch
- apiGroups:
  - ""
  - build.openshift.io
resources:
- buildlogs
verbs:
- get
- list
- watch
- apiGroups:
  - logging.openshift.io
resources:
- elasticsearches
verbs:
- '*'
- apiGroups:
  - logging.openshift.io
resources:
- kibanas
verbs:
- '*'
- apiGroups:
  - packages.operators.coreos.com
resources:
- packagemanifests
verbs:
- '*'
- apiGroups:
  - ""
  - authorization.openshift.io
resources:
- rolebindings
- roles
verbs:

```

```

- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - rbac.authorization.k8s.io
resources:
- rolebindings
- roles
verbs:
- create
- delete
- deletecollection
- get
- list
- patch
- update
- watch
- apiGroups:
  - ""
  - authorization.openshift.io
resources:
- localresourceaccessreviews
- localsubjectaccessreviews
- subjectrulesreviews
verbs:
- create
- apiGroups:
  - authorization.k8s.io
resources:
- localsubjectaccessreviews
verbs:
- create
- apiGroups:
  - ""
  - project.openshift.io
resources:
- projects
verbs:
- delete
- get
- apiGroups:
  - ""

```



```

- authorization.openshift.io
resources:
- resourceaccessreviews
- subjectaccessreviews
verbs:
- create
- apiGroups:
  - ""
  - security.openshift.io
resources:
- podsecuritypolicyreviews
- podsecuritypolicyselfsubjectreviews
- podsecuritypolicysubjectreviews
verbs:
- create
- apiGroups:
  - ""
  - authorization.openshift.io
resources:
- rolebindingrestrictions
verbs:
- get
- list
- watch
- apiGroups:
  - build.openshift.io
resources:
- jenkins
verbs:
- admin
- edit
- view
- apiGroups:
  - ""
  - project.openshift.io
resources:
- projects
verbs:
- delete
- get
- patch
- update
- apiGroups:
  - ""
  - route.openshift.io
resources:
- routes/status

```

```

    verbs:
      - update
  - apiGroups:
      - security.openshift.io
    resourceNames:
      - nonroot-v2
    resources:
      - securitycontextconstraints
    verbs:
      - use
  - apiGroups:
      - ""
    resources:
      - pods/finalizers
    verbs:
      - update

```

```

apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: dr-automation-platform
  namespace: <namespace>
rules:
  - apiGroups: ["" ]
    resources: ["namespaces"]
    resourceNames: ["namespace"]
    verbs: ["get", "list"]
  - apiGroups: ["" ]
    resources: ["pods/exec", "pods", "pods/log"]
    verbs: ["get", "list"]
  - apiGroups: ["batch"]
    resources: ["jobs", "cronjobs"]
    verbs: ["get"]

```

12. Appendix E - Specification

This appendix holds the detailed specification requirements pertaining to each release per system component. Example requirements include, number of replicas, containers detail and memory and core requirements per component.

The information contained in this table appendix, is related to the [High Availability](#) (HA) support sub topic detailed under the Deployment Chapter.

Component	Type	Replicas	Container Name	Container Type	Requests CPU (Cores)	Requests Memory (MB)	Limits CPU (Cores)	Limits Memory (MB)
gitlab-sidekiq-all-in-1-v2	Deployment	1	sidekiq	container	0.2	512	1	2048
			certificates	init	0.2	128	1	1024
			configure	init	0.2	128	1	1024
			dependencies	init	0.2	128	1	1024
gitlab-toolbox	Deployment	1	toolbox	container	0.2	1024	0.2	1024
			certificates	init	0.2	128	1	1024
			configure	init	0.2	128	1	1024
gitlab-webserve-default	Deployment	1	nginx	container	0.1	100	1	1024
			webserve	container	2	2048	2	2048
			gitlab-workhorse	container	0.1	100	1	2500
			certificates	init	0.2	128	1	1024
			configure	init	0.2	128	1	1024
			dependencies	init	0.2	128	1	1024
hasura	Deployment	1	nginx-proxy	container	0.1	128	0.5	1024
			hasura-auth-webhook	container	0.1	128	0.5	1024
			hasura	container	1	2048	2	3072
opensearch	Deployment	1	dashboar	container	1	2048	2	4096

Component	Type	Replicas	Container Name	Container Type	Requests CPU (Cores)	Requests Memory (MB)	Limits CPU (Cores)	Limits Memory (MB)
ch-dashboards	Deployment	1	ch-dashboards	Container	0.1	128	0.25	256
pgadmin4	Deployment	1	pgadmin4-keycloak-gatekeeper	Container	0.2	128	0.35	256
			pgadmin4	Container	0.5	512	0.5	512
postgres-proxy	Deployment	1	stolon	Container	0.5	1024	0.5	1024
postgres-sentinel	Deployment	1	stolon	Container	0.5	1024	0.5	1024
tr-api-gateway	Deployment	1	api-gateway	Container	0.2	128	1	1024
			tr-api-gateway-keycloak-gatekeeper	Container	0.1	128	0.25	256
data-access-service	StatefulSet	1	minio-proxy	Container	0.5	512	2	2048
			data-access-service	Container	0.5	512	4	4096
gitlab-gitaly	StatefulSet	1	gitaly	Container	1	1024	1	1024
			certificates	Init	0.2	128	1	1024
			configure	Init	0.2	128	1	1024
keycloak	StatefulSet	1	keycloak	Container	1	2048	1	2048
			generate-jks	Init	0.2	128	1	1024
logstash-logstash	StatefulSet	1	logstash	Container	1	2048	1	2048
			generate-jks	Init	0.2	128	1	1024

Component	Type	Replicas	Container Name	Container Type	Requests CPU (Cores)	Requests Memory (MB)	Limits CPU (Cores)	Limits Memory (MB)
minio	Statefulset	4	minio	container	1	2048	1	2048
			wait-for-logstash	init	0.1	128	0.25	256
opensearch-cluster-master	Statefulset	1	opensearch	container	1	4096	2	4096
			keystore	init	0.2	128	1	1024
			generate-pkcs8key	init	0.2	128	1	1024
postgres-keeper	Statefulset	1	plasma	container	0.2	1024	1	1024
			stolon	container	8	24576	8	24576
redis-master	Statefulset	1	redis	container	0.1	256	0.5	1024

13. Appendix F - Reporting Database Exposure

13.1. Introduction

The Reporting Database is a dedicated area within ThetaRay's Postgres deployment that includes multiple schemas oriented towards reporting / business intelligence queries. The reporting system issuing queries against the database will typically run outside the boundaries of the Openshift / Kubernetes environment running the Postgres server, requiring the database to be externally exposed.

To facilitate the above, the system supports an optional deployment of 'pgbouncer', a Postgres connection pooler / load balancer in front the internal database server.

This enables:

- Exposure of the database using an external DNS host name
- Using an external certificate / private key for encrypting Postgres traffic through TLS (the certificate will typically be issued through a Certificate Authority already trusted by the BI tool accessing the database)
- Ensuring that only a dedicated user used for reporting purposes can access the database from external clients.

13.2. Configuring the Deployment to Expose the Reporting Database

» To enable Reporting Database exposure ->

1. Under detection-platform/environments/<profile>/common.yaml.gotmpl set:
 - a. postgres_report_user_username: <reporting user name>
 - b. postgres_report_user_password: <reporting user password>
2. postgres_reporting:
 - tls:
 - private cert:
 - a. Cert: <certificate>
 - b. Key: <private key>
 - private_cert_ca:
 - a. Cert: <CA certificate>

As a last step common-values.yaml in the applications / investigation center deployment should contain:

```
postgresExposeReports: false
```

It should be noted that if external DNS is not configured on the environment, the IP address associated with the LoadBalancer service exposing PGBouncer should explicitly be registered in the corporate DNS under the external host name associated with Postgres and the certificate provided in private_cert should be associated with this host as well to ensure that clients can properly verify the connection.

14. Appendix G - Enable SLA in Days

14.1. Introduction

The Display SLA in Days feature allows for the conversion of SLA (Service Level Agreement) display from hours to days. By default, the SLA is displayed in hours, and the feature is disabled.

For our on prem customers who run deployments where it is better suited to display SLA time in days, please follow the instructions detailed below:

14.2. Enablement Process

» To enable the feature:

1. Modify the common-values.yaml file by setting the sla_in_days.enabled flag to true:

```
sla_in_days:true
```

2. This flag is located in the common-values.yaml file, and the default configuration is:

```
1 | sla_in_days:
2 |   enabled: false
```

3. The feature is applied in the deployment through the configmap and is reflected in the config.json file as follows:

```
data:
  config.json: "{\n  \"url\": \"infra-uat.cbtm.mashreqdev.com\", \n  \"realm\": \"thetaray\", \n\n  \n  \"clientId\": \"apps_uat_fe\", \n  \"authorizationClientId\": \"thetaray-authz\", \n\n  \n  \"queryValuesLimit\": \"5000\", \n  \"restrictHtml\": \"false\", \n  \"searchPrefix\" : \"tm_\", \n  \"searchDpv\" : \"\", \n  \"slaInDays\" : \"false\" \n}"
kind: ConfigMap
```

- With reference to the above screenshot change the highlighted "false" to "true"
 - On completion you are required to rescale the tr-icfe Pod
4. Once enabled, the system will display SLAs in days instead of hours.

15. Appendix H - Toggle between CRA 1 and CRA 2

15.1. Introduction

For our on prem customers who have deployed CRA 1 and require to activate version CRA 2, please follow the instructions detailed below:

15.2. CRA 1 - 2 Toggling

» To toggle between CRA versions:

1. in the common-values.yaml file under

```
tr-platform-production.<VERSION>/environments/ocp-prod/common.yaml.gotmpl
```

2. Edit the common.yaml from "phase 1" to "phase 2".

```
cra:
  flavor: phase1
  report_rows_limit_csv: 10000
  report_rows_limit_excel: 10000
  manual_reclassification_notification_target: classification_change_target
```

3. After the change is made run the Shared upgrade again (same version)
4. Once enabled, the system will display CRA 2.

16. Appendix J - Create TM Manual Alert when no Activity

16.1. Introduction

For customers whose deployment needs the flexibility to create TM manual alert when due to non activity there is no Account ID available in the TR system.

Default status - Off (false)

16.2. Instructions

1. Go to Postgres cdd -> schemas -> public -> tables -> feature_toggles.
2. Open a new query page and this query.
3. UPDATE public.feature_toggles.
4. SET enable = true.
5. WHERE tag_name IN ('MVP-54087');
6. In terminal: kubectl get pods -n apps-xxx.
7. Get tr-ICBE pod and copy it.
8. Run new command kubectl delete pod <tr-ICBE-pod> -n apps-xxx.
9. Wait for the new pod will be up and running.

17. Appendix K - Rule Builder & Simulator

For our on prem customers who deploy Rule Builder & Simulator, the following information will be of interest:

17.1. Deployment

- common.yaml
- rule builder request /response timeout - default 30 seconds

```
1 | rule_builder:  
2 |   httpx_timeout: 30
```