Web Services and Applications Deployment Guide

Observability

4/4/2025

Contents

Observability

Genesys Web Services 8.6 provides deployment instructions and scripts which you can use when installing and configuring Observability based on Grafana's open source software solution.

The benefits of using this Observability Solution are:

1. Help with root cause analysis.
2. Help with resilience and reliability (MTTR, uptime, and so on).
3. Help with faster and more informed feedback loops (think: continuous improvement).
4. Provide centralized and searchable storage for logs and metrics for Genesys Web Services 8.6.
5. Enable customers to directly share relevant logs and metrics with Genesys Customer Support to help with troubleshooting during incidents happening in an on-premise environment.

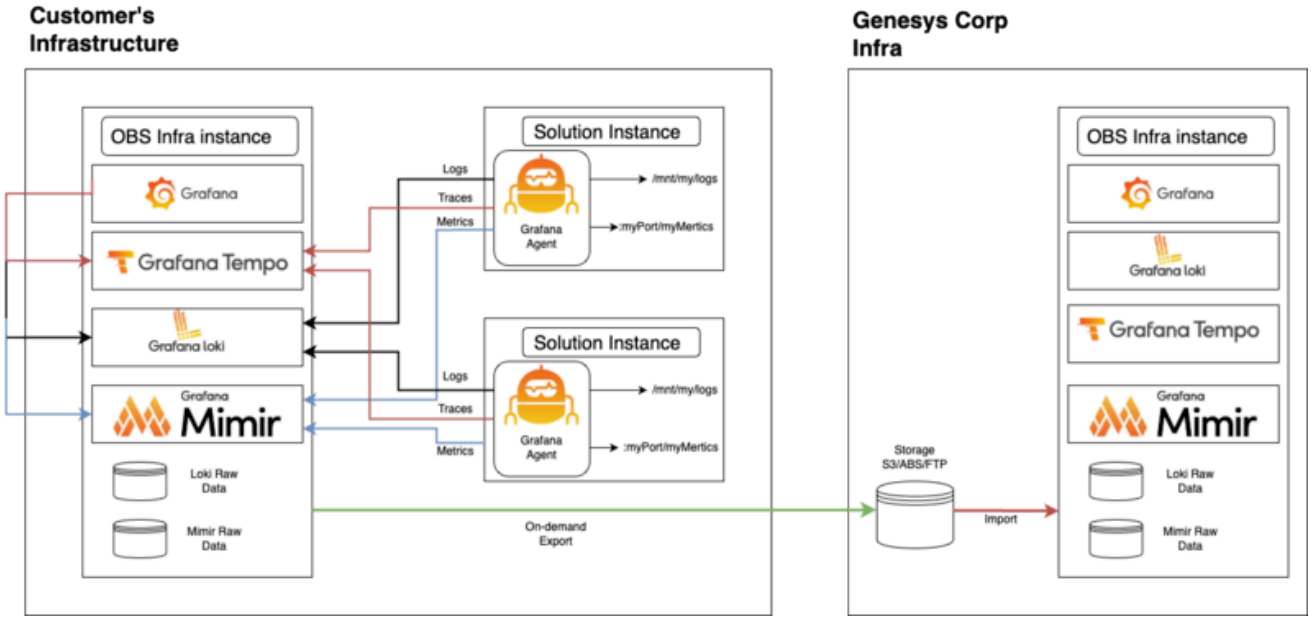
The Observability Solution is founded upon Grafana’s Observability Stack (open source) which consists of the following components:

- **Grafana** - analytics and visualization
- **Mimir** - storage for metrics
- **Loki** - log aggregation backend
- **Tempo** – tracing backend
- **Grafana Agent** - vendor-neutral telemetry collector

Architecture

The Observability Solution architecture consists of:

- OBS Core Stack (Grafana, Loki, Mimir, Tempo) running on a single designated instance.
- Grafana Agents configured on instances where Genesys’ software is running and sends data to the OBS Core Stack.
- On-demand export from OBS Core Stack to Genesys (the nature of "On-demand" is completely controlled by customer).



Observability platform installation

> **Important**
>
> Genesys has provided deployment instructions and scripts which install the Grafana platform with Genesys Web Services 8.6. However, Genesys cannot provide support for any issues encountered with the Grafana platform.

> **Important**
>
> In order to ensure you are using the latest instructions and to gain access to the RPM package referenced below, refer to the Observability installation package included in Genesys Web Services 8.6.
>
> **OBS_instance/README.md**

Installation procedure

Complete the following steps to install Grafana.

1. Get the repository keys.

```
wget -q -O gpg.key https://rpm.grafana.com/gpg.key
```

```
sudo rpm --import gpg.key
```

2. Create **/etc/yum.repos.d/grafana.repo** with the following parameters.

```
[grafana]
name=grafana
baseurl=https://rpm.grafana.com
repo_gpgcheck=1
enabled=1
gpgcheck=1
gpgkey=https://rpm.grafana.com/gpg.key
sslverify=1
sslcacert=/etc/pki/tls/certs/ca-bundle.crt
```

3. Install packages.

```
sudo dnf install grafana-enterprise loki mimir tempo
```

4. Update the loki configuration file **(/etc/loki/config.yml)** with the following parameters:

```
# /etc/loki/config.yml
auth_enabled: false
server:
  http_listen_port: 3100
  grpc_listen_port: 9096
  grpc_server_max_recv_msg_size: 10000000
  grpc_server_max_send_msg_size: 10000000
common:
  instance_addr: 127.0.0.1
  path_prefix: /tmp/loki
  storage:
    filesystem:
      chunks_directory: /tmp/loki/chunks
      rules_directory: /tmp/loki/rules
  replication_factor: 1
  ring:
    kvstore:
      store: inmemory
query_range:
  results_cache:
    cache:
      embedded_cache:
        enabled: true
        max_size_mb: 100
schema_config:
  configs:
    - from: 2022-11-30
      store: tsdb
      object_store: filesystem
      schema: v12
      index:
        prefix: index_
        period: 24h

ruler:
  alertmanager_url: http://localhost:9093
#### custom part
compactor:
  working_directory: /tmp/loki/retention
  shared_store: filesystem
  compaction_interval: 10m
  retention_enabled: true
  retention_delete_delay: 2m
```

```
    retention_delete_worker_count: 150
  limits_config:
    retention_period: 7d
    ingestion_rate_mb: 7
    ingestion_burst_size_mb: 20
```

5. Update the Mimir configuration file **(/etc/mimir/config.yml)** with the following parameters:

```
# /etc/mimir/config.yml

multitenancy_enabled: false
target: all,alertmanager,overrides-exporter
server:
  http_listen_port: 9009
  # Configure the server to allow messages up to 100MB.
  grpc_server_max_recv_msg_size: 104857600
  grpc_server_max_send_msg_size: 104857600
  grpc_server_max_concurrent_streams: 1000
ingester:
  ring:
    # The address to advertise for this ingester.  Will be autodiscovered by
    # looking up address on eth0 or en0; can be specified if this fails.
    instance_addr: 127.0.0.1
    # We want to start immediately and flush on shutdown.
    min_ready_duration: 0s
    final_sleep: 0s
    num_tokens: 512
    # Use an in memory ring store, so we don't need to launch a Consul.
    kvstore:
      store: memberlist
    replication_factor: 1
blocks_storage:
  tsdb:
    dir: /tmp/mimir/tsdb
  filesystem:
    dir: ./data/tsdb
compactor:
  data_dir: /tmp/mimir/compactor
ruler_storage:
  backend: local
  local:
    directory: /tmp/mimir/rules
##### custom part
limits:
  compactor_blocks_retention_period: 2w
  max_global_series_per_user: 0
  ingestion_rate: 1000000
  ingestion_burst_size: 20000000
memberlist:
  join_members: [ localhost ]
```

6. Update the Tempo configuration file **(/etc/tempo/config.yml)** with the following parameters.

```
#/etc/tempo/config.yml
server:
  http_listen_port: 3200
  grpc_listen_port: 9097
query_frontend:
  search:
    duration_slo: 5s
    throughput_bytes_slo: 1.073741824e+09
  trace_by_id:
    duration_slo: 5s
```

```
distributor:
  receivers:
    jaeger:
      protocols:
        thrift_http:
        grpc:
        thrift_binary:
        thrift_compact:
    zipkin:
    otlp:
      protocols:
        http:
        grpc:
    opencensus:

ingester:
  max_block_duration: 5m
  #enable_inet6: false

compactor:
  compaction:
    block_retention: 24h
  #enable_inet6: false

metrics_generator:
  registry:
    external_labels:
      source: tempo
      cluster: docker-compose
    #enable_inet6: false
  storage:
    path: /tmp/tempo/generator/wal
    remote_write:
      - url: http://localhost:9009/api/v1/write
        send_exemplars: true
storage:
  trace:
    backend: local
    wal:
      path: /tmp/tempo/wal
    local:
      path: /tmp/tempo/blocks
overrides:
  defaults:
    global:
      max_bytes_per_trace: 10000000
    metrics_generator:
      processors: [service-graphs, span-metrics] # enables metrics generator
```

7. Create Grafana's default data source file **(/etc/grafana/provisioning/datasources/grafana-provisioning-datasources.yaml)** with the following parameters:

```
# goes to /etc/grafana/provisioning/datasources/grafana-provisioning-datasources.yaml
apiVersion: 1
datasources:
  - name: Mimir
    uid: mimir
    type: prometheus
    access: proxy
    orgId: 1
    url: http://localhost:9009/prometheus
    version: 1
    editable: true
```

```
    jsonData:
      httpHeaderName1: 'X-Scope-OrgID'
      alertmanagerUid: 'alertmanager'
    secureJsonData:
      httpHeaderValue1: 'demo'
    isDefault: true
  - name: Mimir Alertmanager
    uid: alertmanager
    type: alertmanager
    access: proxy
    orgId: 1
    url: http://localhost:9009/
    version: 1
    editable: true
    jsonData:
      httpHeaderName1: 'X-Scope-OrgID'
      implementation: 'cortex'
    secureJsonData:
      httpHeaderValue1: 'demo'
  - name: Loki
    uid: loki
    editable: true
    type: loki
    access: proxy
    url: http://localhost:3100
    jsonData:
      derivedFields:
        - datasourceUid: tempo
          matcherRegex: "traceId\":\"(\\w+)\""
          name: traceId
          url: '$${__value.raw}'
          urlDisplayLabel: ''
        - datasourceUid: tempo
          matcherRegex: "trace_id\":\"(\\w+)\""
          name: trace_id
          url: '$${__value.raw}'
          urlDisplayLabel: ''
      timeout: 125
  - name: Tempo
    uid: tempo
    editable: true
    type: tempo
    access: proxy
    url: http://localhost:3200
```

8. Update the file **/etc/grafana/provisioning/dashboards/sample.yaml** with the following parameters:

```
# /etc/grafana/provisioning/dashboards/sample.yaml
apiVersion: 1
providers:
 - name: 'default'
   orgId: 1
   folder: ''
   folderUid: ''
   type: file
   options:
     path: /var/lib/grafana/dashboards
```

9. Create a folder **/var/lib/grafana/dashboards**.
10. Copy attached dashboards files from the path: **OBS_instance/dashboards** to **/var/lib/grafana/dashboards**.

11. Copy attached recording rules to **/tmp/mimir/rules/anonymous**.
12. Enable and start services.

```
systemctl enable loki
systemctl start loki
systemctl enable mimir
systemctl start mimir
systemctl enable tempo
systemctl start tempo
systemctl enable grafana-server
systemctl start grafana-server
```

13. Check the statuses of the newly deployed services.

```
systemctl status loki
systemctl status mimir
systemctl status tempo
systemctl status grafana-server
```

14. Check the accessibility of Grafana UI using the following URL and update the default password:
    https://your_host_ip:3000/

Grafana Agent installation

The following steps must be repeated for any host which has Genesys Web Services 8.6 deployed.
grafana_agent_static/README.md

1. Copy the **.tar.gz** package to the nodes where the agent needs to be installed.
2. Untar the archive.
3. From the **grafana_agent_static/config** folder, run the **agent_install.sh** script with the following two arguments.
    1. the hostname of Loki/Mimir/Grafana instance
    2. the path to folder with GWS logs location, that is, /logs:

```
ls -l /logs
total 16
drwxrwxr-x 2 genesys genesys 16384 Mar  5 14:20 gws-api-v2
drwxrwxr-x 2 genesys genesys    38 Mar  5 11:28 gws-service-platform
```

4. Script should be executed with root permissions, that is, **sudo ./agent_install.sh <OBS_HOSTNAME_OR_IP> <LOG_FOLDER>**

For example, **sudo ./agent_install.sh d-pat-u1040.us.int.genesyslab.com /mnt/logs/dir_with_logs_from_gws_components/**