What's New in EF Portal 2025.1

info@ni-sp-software.com / www.ni-sp-software.com

Table of Contents

EF Portal Monitoring to StatsD / Grafana

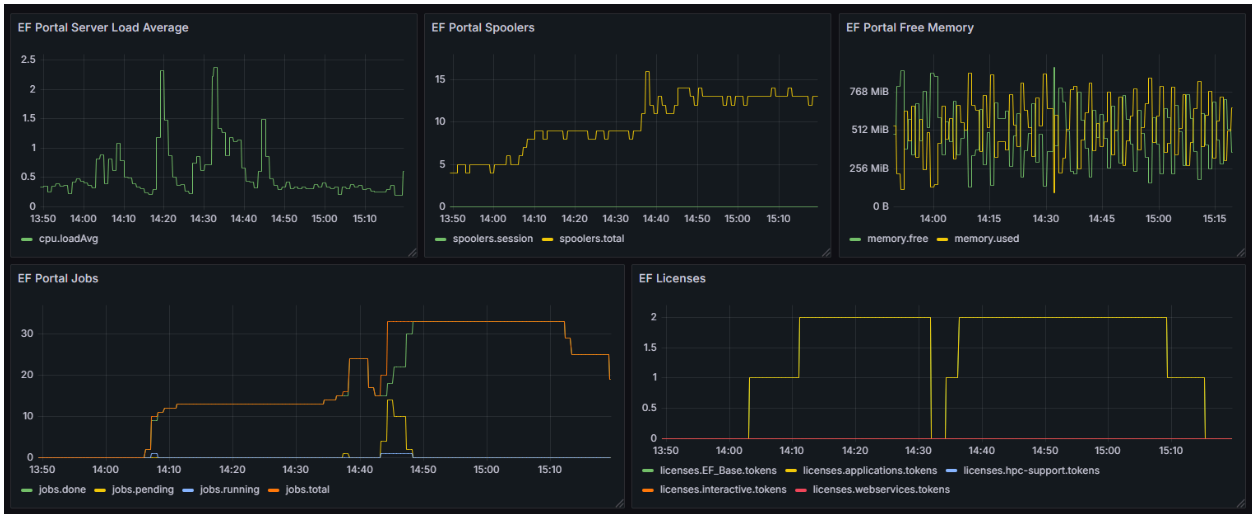
The new monitoring into StatsD offers tracking a range of metrics of EF Portal into the modern monitoring StatsD aggregator including support for visualization in Grafana and other tools.

We track the following metrics which can be stored into a StatsD server typically owned by the customer for further postprocessing in e.g. InfluxDB, Graphite and Grafana:

- stats.gauges.cpu.loadAvg
- stats.gauges.jobs.done
- stats.gauges.jobs.pending
- stats.gauges.jobs.running
- stats.gauges.jobs.total
- stats.gauges.licenses.EF_Base.tokens
- stats.gauges.licenses.applications.tokens
- stats.gauges.licenses.hpc-support.tokens
- stats.gauges.licenses.interactive.tokens
- stats.gauges.licenses.webservices.tokens
- Stats.gauges.licenses.rest.tokens
- stats.gauges.filesystem.Sessi.fsAvailInodes
- stats.gauges.filesystem.Sessi.fsAvailSize
- stats.gauges.filesystem.Sessi.fsUsedInodes
- stats.gauges.filesystem.Sessi.fsUsedSize
- stats.gauges.filesystem.Sessi.usedSize
- stats.gauges.memory.free
- stats.gauges.memory.used
- stats.gauges.spoolers.session
- stats.gauges.spoolers.total
- stats.gauges.statsd.timestamp_lag
- Stats.gauges.users.logged

The built-in RRD database continues to support immediate administrator views of different metrics of EF Portal. For long term storage and post-processing we recommend to use the new StatsD integration.

In Grafana this can look similar to this:



Sending StatsD metrics can be enabled by configuring

- `STATSD_ADDR    # e.g. statsd.example.com`
- `STATSD_PORT    # e.g. 1825`

in $EF_ROOT/plugins/admin/conf/admin.statistics.efconf.

Session Pre-Submit and Closing Hook

Pre-Submit and Closing hooks are extension points to support executing custom actions before the session is actually submitted and after a session is closed. With a pre-submit hook e.g. a instance or VM can be instantiated or a server can be powered on. In the closing hook the resource running the session can then be shutdown or turned off again to free resources and save cost.

Hook script location are configured using the parameters INTERACTIVE_SESSION_PRE_SUBMIT_HOOK and INTERACTIVE_SESSION_CLOSING_HOOK in either:

- $EF_ROOT/plugins/interactive/conf/interactive.efconf
- $EF_CONF_ROOT/plugins/interactive/interactive.efconf

Pre-submit hooks are not blocking: they are asynchronously executed by a trigger when the session is in Pending state. If the pre-submit hook script fails, the session is not submitted and goes in Failed state.

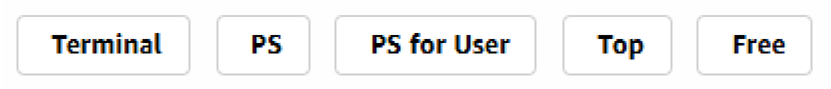
Like other hooks, the configured hook script is executed by the user running the EF Portal Server. The scripts can be put in any location, as long as it is readable by the Server. Hook script logs are written into the session spooler and is available for reading in the "Details" -> "Session Debug Info" panel for the corresponding session.

Environment variables can be shared between hook scripts to enable execution correlation. Any exported variable which starts with the SESSION_ prefix is automatically passed along hook scripts execution, thanks to session spooler metadata.

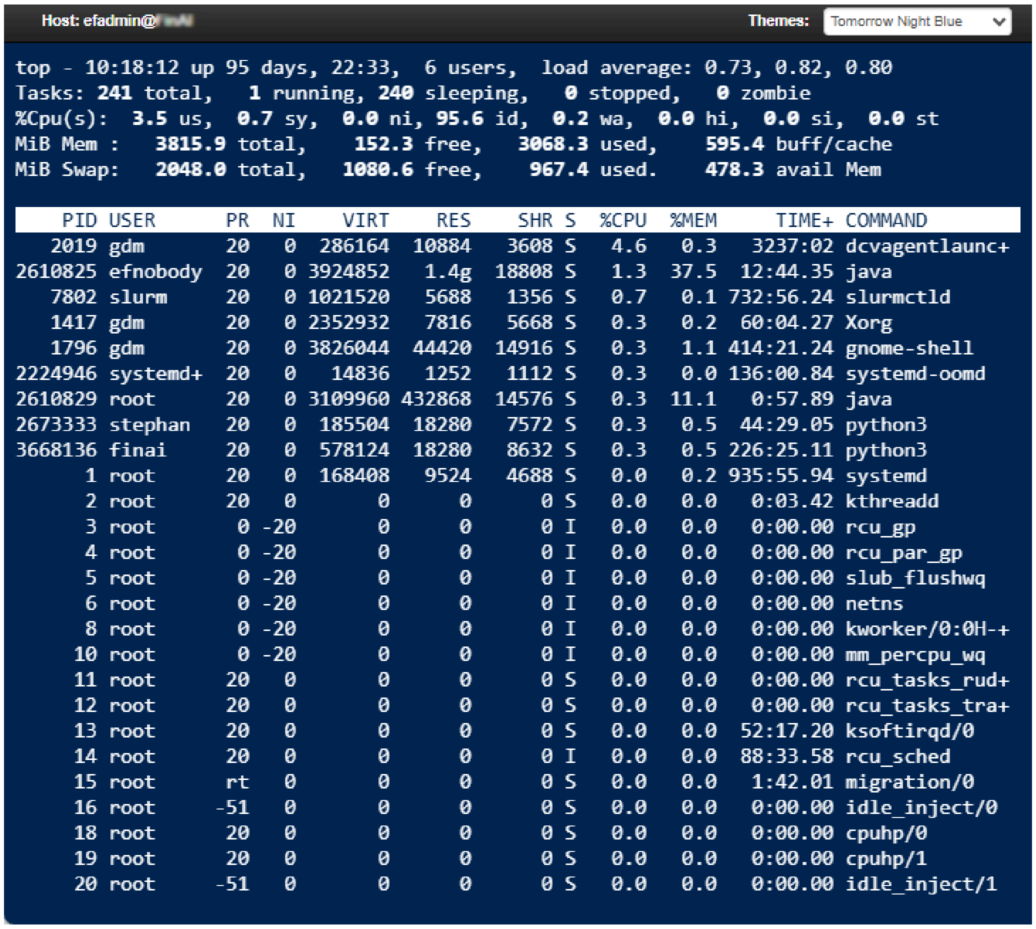
Sample scripts for pre-submit and closing hooks can be found in $EF_ROOT/plugins/interactive/bin/samples. E.g. sample.session.pre.submit.hook.sh and sample.session.closing.hook.sh. Another hook available is the INTERACTIVE_SESSION_STARTING_HOOK.

WebTerminal

The new WebTerminall gives administrators immediate access to remote hosts in the cluster from the EF Portal hosts view. In addition to the quick commands we introduced in EF Portal 2025.0 the new “Terminal” functionality as highlighted in the following screenshot from the EF Portal Hosts View:



When you click on Terminal a new browser window opens with the Terminal where you can login into a remote shell:



```
top - 10:18:12 up 95 days, 22:33,  6 users,  load average: 0.73, 0.82, 0.80
Tasks: 241 total,   1 running, 240 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.5 us,  0.7 sy,  0.0 ni, 95.6 id,  0.2 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  3815.9 total,   152.3 free,  3068.3 used,   595.4 buff/cache
MiB Swap:  2048.0 total,  1080.6 free,   967.4 used.   478.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2019 gdm       20   0  286164  10884   3608 S   4.6   0.3   3237:02 dcvagentlaunc+
2610825 efnobody  20   0 3924852   1.4g  18808 S   1.3  37.5  12:44.35 java
   7802 slurm     20   0 1021520   5688   1356 S   0.7   0.1 732:56.24 slurmctld
   1417 gdm       20   0 2352932   7816   5668 S   0.3   0.2  60:04.27 Xorg
   1796 gdm       20   0 3826044  44420  14916 S   0.3   1.1 414:21.24 gnome-shell
2224946 systemd+  20   0   14836   1252   1112 S   0.3   0.0 136:00.84 systemd-oomd
2610829 root      20   0 3109960 432868  14576 S   0.3  11.1   0:57.89 java
2673333 stephan   20   0  185504  18280   7572 S   0.3   0.5  44:29.05 python3
3668136 finai     20   0  578124  18280   8632 S   0.3   0.5 226:25.11 python3
      1 root      20   0  168408   9524   4688 S   0.0   0.2 935:55.94 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:03.42 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_flushwq
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-+
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     11 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_rud+
     12 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_tra+
     13 root      20   0       0      0      0 S   0.0   0.0  52:17.20 ksoftirqd/0
     14 root      20   0       0      0      0 I   0.0   0.0  88:33.58 rcu_sched
     15 root      rt   0       0      0      0 S   0.0   0.0   1:42.01 migration/0
     16 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     18 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     19 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     20 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
```

A number of themes selectable from the upper right pulldown offer adapting the look and feel of the web terminal on the host.

Configuration of the WebTerminal is done via the environment configured in e.g. /opt/nisp/enginframe/conf/shell:

```
# Duration of inactivity (in milliseconds) after which the remote
shell connection will be closed.
# Default: 300,000 milliseconds (5 minutes).
# SHELL_INACTIVE_TIMEOUT_MS=300000

# Maximum allowed duration (in milliseconds) for a remote shell
connection.
# Default: 3,600,000 milliseconds (60 minutes).
# SHELL_MAX_DURATION_MS=3600000

# Command used to establish the remote shell connection to the host
(e.g., rsh).
# Default: ssh
# SHELL_SSH_WRAPPER=ssh
# SHELL_SSH_WRAPPER="srun --pty --time=60 --nodelist=%NODE%
/bin/bash"
# SHELL_SSH_WRAPPER="lsrun -m %NODE% /bin/bash"

# A colon (:) separated list of hostnames that defines the accessible
hosts in the cluster.
# Example: host01:host03:host05
# Default: Empty list
# SHELL_SSHHOST_ALLOWLIST=

# Path to your private key file used for securing HTTP connections
(HTTPS).
SHELL_PRIVATE_KEY=/opt/nisp/enginframe/conf/shell/certs/shell.key.pem

# Path to your certificate file used for securing HTTP connections
(HTTPS).
SHELL_CERTIFICATE=/opt/nisp/enginframe/conf/shell/certs/shell.cert.pe
m

# Port number on which the EF Portal Remote Shell Server listens for
incoming connections.
# Default: 3000
SHELL_PORT=3000
```

The remote host can be accessed in the WebTerminal via ssh or interactive scheduler commands like srun or lsrun.

The logfiles of the WebTerminal can be found in the files ef.shell.stdout and ef.shell.stderr in the standard EF Portal log directory (eg. /opt/nisp/enginframe/logs/$HOSTNAME/ef.shell.std*).

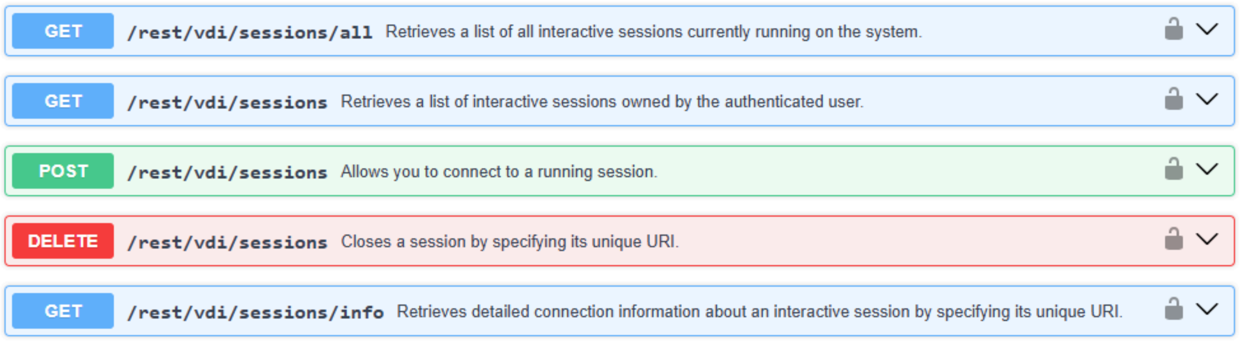
REST API Endpoints for VDI Management

In addition to the large amount of REST API endpoints (overview can be found in EFPURL/enginframe/docs/rest/swagger.html) we added endpoints to manage EF Portal VDI functionality:

- Get Session Information: GET /rest/vdi/sessions: list your sessions based on *filter* and *query* parameters;
- Connect: POST /rest/vdi/sessions: retrieve session connection URL or Connection File based on *uri* and *type* parameters.
- Show all Sessions: GET /rest/vdi/sessions/all: [ADMINS ONLY] list all sessions based on *filter* and *query* parameters;

- Session Information: GET /rest/vdi/sessions/info: retrieve session connection details based on *uri* parameter;
- Close Session: DELETE /rest/vdi/sessions: close session based on *uri* parameter;

Overview of VDI REST API endpoints in the Swagger UI where you can try those endpoints directly or get the related curl commands:



GET /rest/vdi/sessions/all Retrieves a list of all interactive sessions currently running on the system.

GET /rest/vdi/sessions Retrieves a list of interactive sessions owned by the authenticated user.

POST /rest/vdi/sessions Allows you to connect to a running session.

DELETE /rest/vdi/sessions Closes a session by specifying its unique URI.

GET /rest/vdi/sessions/info Retrieves detailed connection information about an interactive session by specifying its unique URI.

REST API Output JSON converted from XML

In addition to showing XML output from the REST API we automatically output JSON converted from XML in the REST API Output for easy consumption in other services. Here is an example of calling a service to get the connection file data which is by default XML:

```
curl --silent -k -X 'POST' \
  'https://efp.example.com:8443/enginframe/rest/system/services' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer
9a15aa62447f4e5dc8b23068a55c6f6ab9ec199e'\
  -H 'Content-Type: application/json' \
  -d '{"sdf": "/hydrogen/ui.hydrogen.xml", "uri":
"//ui.hydrogen/interactive.connection.data", "options": [{ "name":
"sessionUri", "values":
["spooler:///opt/nisp/enginframe/spoolers/efadmin/tmp1080141928575131
6539.session.ef"]}]}' \
| jq -r .outputJson | jq
```

Example output of the JSON output of the connection file data service:

```
{
  "connection": {
    "param": [
      {
        "name": "connectFileUri",
        "content": "//com.enginframe.interactive/generate.dcv2file"
      },
      {
        "name": "connectFileExt",
        "content": ".dcv"
      },
      {
        "name": "uriScheme",
        "content": "https"
      }, ……………
```

Other Features and Improvements

We have added a number of other enhancements:

- Improved Drag & Drop upload
- Improved service editor locking
- Updated logic to prevent quick clicks on session start function
- Improved token management in Jupyter Notebook service
- Remove legacy Neutro code in EF Portal
- Startup time of EF Portal reduced by about 40%
- Number of other performance improvements
- REST /monitor endpoints do not consume a REST license
- Replace BSF/BSH with Groovy

Security / Upgrades

- Upgraded Apache Tomcat to version 9.0.107
- Upgraded Hazelcast to version 5.3.5
- Upgraded Quartz to version 2.5.0
- Upgraded Apache CXF to version 3.5.11
- Upgraded Apache Commons Lang to version 3.17.0
- Upgraded Apache Commons Text to version 1.13.1
- Upgraded Apache Commons IO to version 2.19.0
- Upgraded Apache Commons Logging to version 1.3.5
- Upgraded Apache Commons Codec to version 1.18.0
- Upgraded Apache Commons JEXL to version 2.1.1
- Upgraded Jackson to version 2.13.5
- Upgraded Google Guava to version 32.0.1-jre
- Upgraded Google Guava FailureAccess to version 1.0.3
- Upgraded Google Gson to version 2.13.0

- Upgraded JFreeChart to version 1.0.19
- Upgraded Yauaa to version 7.30.0
- Upgraded OWASP Java Encoder to version 1.3.1
- Upgraded HyperSQL Database to version 2.7.4
- Upgraded C3P0 to version 0.10.2
- Upgraded Mchange Commons Java to version 0.3.2