Ellipse APM On-Premise Installation Guide

Document availability: Public

Document type: Installation Guide

December 10th, 2018

Legal Disclaimer the product described in this documentation may be connected to, and/or communicate information and data via, a network interface, which should be connected to a secure network. It is your sole responsibility to ensure a secure connection to the network and to establish and maintain appropriate measures (such as but not limited to the installation of firewalls, application of authentication measures, encryption of data, installation of antivirus programs, etc.) to protect the product, the network, your systems, and the interface against any kind of security breach, unauthorized access, interference, intrusion, leakage, damage, or corruption or theft of data. We are not liable for damages or losses related to any such security breach, unauthorized access, interference, intrusion, leakage, damage, or corruption or theft of data.

Contents

Ellipse APM On-Premise Prerequisites

Ellipse APM 5.1 is using **Microservices Orchestration Platform** platform as a provider of environment for Microservices. **Microservices Orchestration Platform** 1.0.1 should be finished.

Microservices Orchestration Platform Requirements

Minimum certified Microservices Orchestration Platform cluster should contain:

- Kubernetes Master Node
- One Kubernetes node)
- One Kafka node
- One node for monitoring tools (like Grafana) and logging services (Elastic Search, FileBeat, Kibana)

After Microservices Orchestration Platform cluster is installed following installation artifacts should be prepared: - Kubernetes Command Line Interface configuration file (also known as kubeconfig) - (optionally) Kubernetes Dashboard access token. This is not required to install APM, however it's highly recommended to verify if installation was successful.

Installation machine Installation of Ellipse APM is based on PowerShell scripts. To be able to successfully setup Ellipse APM installation machine should have:

- Access to SQL Server on which Ellipse APM will be created on (TCP port)
- Access to internet (optional - not required if Kubernetes Command Line Interface will be provided differently)
- Access to management port on Microservices Orchestration Platform cluster (By default port 6443 on Master node IP should be accessible)

Windows SQL server installed

- Mixed - authentication enabled
- TCP connection enabled on firewall (for database port)

Active Directory created and setup for users for Ellipse APM

- Two groups should be create on Active Directory
    1. Group with Engineer role on Ellipse APM
    2. Group with Administrator role on Ellipse APM

Active Directory Federation Services installed and configured to communicate with Active Directory On-Premise installation of Ellipse APM is using AD FS service on Windows Server machines. This service should be configured to use Active Directory which contain Ellipse APM users.

Map provider service Ellipse APM requires Map Provider service for some of its functionalities. Currently supported and certified Map Providers Services are: - Bing maps - OpenStreet maps

Configuration of Map services should be included into Customer definition JSON file Customer.json explained

Verify if all prerequisites below are met

1. Configuring Active Directory Management user
2. Managing Ellipse APM database
3. Configuring Active Directory with Active Directory Federation Services
4. Downloading Kubernetes Command Line Interface
5. Configuring HA Proxy
6. Configuring Map Provider Service

Configuring Active Directory Management user

Prerequisites

1. AD Configured with LDAP services enabled
2. SSL configuration enabled on AD server
3. User with permissions to read other users data should be created on the AD.

Following parameters should be prepared for installation:

Parameters	Descriptions
LdapServerAddress	Hostname of Active Directory server used for User management in Ellipse APM

LdapServerPort	Port opened on Active Directory. By default for Windows the value is 389
UseSecureSocketLayer	Flag to use secure layer connection for LDAP communications
LdapFetchDataUserLogin	Name of the user that has rights to fetch details about other users configured in AD
LdapFetchDataUserPassword	Password for user that has rights of fetch details about other users configured in AD
LdapTlsCertificate	Self-signed certificate exported into BASE64 format (Without whitespaces). Not required for normal certificates

WindowsActiveDirectoryRolesJsonDictionary	Dictionary of groups and roles. It should contain mapping between Groups on Active Directory server and Role in Ellipse APM. Roles currently available in APM are: Administrator, Engineer. See example below how to create specified JSON. **Important** Only one group with role Administrator and one group with role Engineer can be specified.

Constructing Windows Active Directory Json

To create proper JSON string we need distinguished name of group on Active Directory that will contain users with Administrator permissions and distinguished name of group on Active Directory that will contain users with Engineer permissions.

Getting distinguish group name:

1. Login to Active Directory Server
2. Open PowerShell Console
3. Execute: Get-ADGroup -Identity <ActiveDirectoryGroupName>

Sample execution for **ApmAdminstrators** (name is only an example)

```
PS C:\Users\Administrator> Get-ADGroup -Identity ApmAdministrators
```

DistinguishedName : CN=ApmAdministrators,CN=Users,DC=ad,DC=dev,DC=apm,DC=somesoftware,DC=abb
GroupCategory : Security
GroupScope : Global
Name : ApmAdministrators
ObjectClass : group
ObjectGUID : 672301ac-d552-sds-b5e8-20ad0b8804f8
SamAccountName : ApmAdministrators
SID : S-1-5-21-70064sdsd5496-1043318291-3401593781-1103

Having both distinguished name for Administrator and for Engineer we can construct a Json string

Note: Please remember to escape every quote symbol in the string (as done in example above)

{\"CN=ApmEngineers,CN=Users,DC=qa-adfs,DC=dev,DC=apm,DC=somesoftware,DC=abb\":\"Engineer\",\"CN=ApmAdministrators,CN=Users,DC=qa-adfs,DC=dev,DC=apm,DC=somesoftware,DC=abb\":\"Administrator\"}

Constructing Customer JSON LDAP configuration

Having all information mentioned above a new section in Customer Json file should be created. Following pattern below:


```
{
  "LdapConfiguration": {
    "LdapServerAddress": "qa-adfs.dev.apm.somefotware.abb",
    "LdapServerPort": "389",
    "UseSecureSocketLayer": true,
    "LdapFetchDataUserLogin": "domainName\\ldapsuer",
    "LdapFetchDataUserPassword": "ldappassword",
    "LdapTlsCertificate": null,
    "WindowsActiveDirectoryRolesJsonDictionary": "{\"CN=ApmEngineers,CN=Users,DC=qa-adfs,DC=dev,DC=apm,DC=somesoftware,DC=abb\":\"Engineer\",\"CN=ApmAdministrators,CN=Users,DC=qa-adfs,DC=dev,DC=apm,DC=somesoftware,DC=abb\":\"Administrator\"}"
  }
}
```


Managing Ellipse APM database

Prerequisites

- Downloaded installation package to installation machine, unzipped into

- PowerShell Console (version at least 4.0)
- Database port should be accessible installation machine
- Customer JSON file created and saved into See: Customer.json explained

Creating/updating database

Prepare required information for script execution

Parameter Name	Description
SQL_CONNECTION_STRING	Connection string to SQL Server on which database exists or should be created.
CUSTOMER	Name of tenant that will be configured on Ellipse APM instance. Database for specified will be created if it doesn't already exist
DATABASE_ACTION	Available actions: "update", "delete"
DATABASE_TOOL_PATH	Path to database management tool. ABB.APM.DatabaseManagementTool.App.dll should be available under this location

Example variable definition:

```
$DATABASE_TOOL_PATH="C:\InstallationPackage\apm-buildingblocks-databasemanagementtool\database-management-tool"
$SQL_CONNECTION_STRING="Data Source=172.0.0.11,50234;User ID=sa;Password=DifficultPassword1; "
$CUSTOMER="TestTenant"
$DATABASE_ACTION="update"
```

Run command:

```
<INSTALLATION_DIR>\apminstall\InstallationPackage\InstallScripts\DatabaseManagement.ps1 -DatabaseToolPath $DATABASE_TOOL_PATH -SqlConnectionString $SQL_CONNECTION_STRING -Customer $CUSTOMER -databaseAction $DATABASE_ACTION
```

Note: Database Management tool can create a database on SQL server if specified database doesn't exists.

Connection string for created database will be looking like this:
$SQL_CONNECTION_STRING;Initial Catalog=AHC-$CUSTOMER

Verifying database connections

1. Quickest was to verify database connection is opening a database using connection string provided above.

2. If database was successfully created than there should be tables and view created.

Troubleshooting

1. Verify if Dynamic TCP port is opened to Kubernetes cluster and Database Management Tool
2. Verify that tables were created on database
3. Verify Database management tools logs to see if the migration process was successful

Configuring Active Directory with Active Directory Federation Services

Prerequisites

- Copy apminstall to machine that will be used as Active Directory Federation Services server
- Active Directory services are available from AD FS machine.
- PowerShell Console (version at least 4.0)

Fresh setup of AD FS

1. Verify that role ADFS is enabled on Windows Machine
2. Open Server Manager and Select **AD FS** role on Overview dashboard.
3. Verify that **AD FS** has an **AD** server configured properly (at least one server should be online)
4. Prepare following parameters

Parameter Name	Description
APP_GROUP_ID	Name for application group in AD FS, all created roles and service principals will be created under this name. *String Value*
REDIRECT_URLS	List of URLs to which APM will be redirected to.
ENGINEERS_GROUP_ID	ID of Active Directory group in which user of type "Engineers". ID can be taken using the instruction below (or received from Active Directory Administrator)
ADMIN_GROUP_ID	ID of Active Directory group in which user of type "Administrators". ID can be taken using the instruction below (or received from Active Directory Administrator)

Example of parameters definition

```
$APP_GROUP_ID = "APM-QA"
$REDIRECT_URLS="http://my-apm.enterprisesfotware.com/signin-oidc"
$ENGINEERS_GROUP_ID = "CS-ID-WoRk-Ers"
$ADMIN_GROUP_ID = "S-ID-WoRk-Ers"

.\PrincipalsCreator-ADFS.ps1 -AppGroupID $APP_GROUP_ID -WebserviceRedirectUri $REDI
RECT_URLS -AdminGroup $ADMIN_GROUP_ID -EngineersGroup $ENGINEERS_GROUP_ID
```

Successful execution should print results:

```
---------------------------------------------------------------
Web Service API
ClientId:    apm-webservice-app
ClientSecret: svEFtwKSwo0c3QnYKqhawAfn2u9SYqPUd3s53sNM
---------------------------------------------------------------
---------------------------------------------------------------
apm-webservice-superadmin
ClientId:    apm-webservice-superadmin
ClientSecret: yXtE3E_yOyCiJ1daUBB0WT8cAMSPriZwixMa9LPA
---------------------------------------------------------------
---------------------------------------------------------------
apm-webservice-scheduler
ClientId:    apm-webservice-scheduler
ClientSecret: 1HOhgf53B_TYuu8t-kjkCWF35hIODpIM0ka-0jhW
---------------------------------------------------------------
---------------------------------------------------------------
apm-webservice-import
ClientId:    apm-webservice-import
ClientSecret: 2v6X66A9JGbYauPV15AcAXhbDHl7rPPKLEey-Xx4
---------------------------------------------------------------
---------------------------------------------------------------
apm-webservice-administrator
ClientId:    apm-webservice-administrator
ClientSecret: 2v6X666asdasduPV1AcAXhbDHl7rPPKLEey-Xx4
---------------------------------------------------------------
```

Save this results in secure place as this will be used for configuring Ellipse APM solution.

How to get ID of a group in Windows AD

1. Login to Windows AD
2. Open PowerShell
3. Execute command:

```
Get-ADGroup -Identity <AD_GROUP_NAME>
```

Example:

DistinguishedName : CN=ApmAdmins,CN=Users,DC=enterprisesoftware,DC=abb
GroupCategory : Security
GroupScope : Global
Name : ApmAdmins
ObjectClass : group
ObjectGUID : 1710e869-378b-4b92-9fcb-6c24c5838d53
SamAccountName : ApmAdmins
SID : S-1-5-21-1991648568-3821725637-303003404-1101

How to get new Secret Keys for Service Principals generated on AD FS

1. In Server Manager click **Tools** > **AD FS Management**
2. Click **Application Groups**
3. Find group with name. Click on it and select **Properties** from Actions pane
4. For each Server Application on the list. Click on the element on the list. Click **Edit...**
5. In new window click **Confidential** tab and click button **Client secret**
6. Copy generated values and save it for later

Verifying if AD FS server works properly

1. Go to apminstall
2. Execute following scripts using data generated in previous steps.

```
.\GetAuthenticationToken.ps1 -clientId apm-webservice-superadmin -clientKey yXtE3E_yOyCiJ1daUBB0WT8cAMSPriZwixMa9LPA -resourceAppId apm-webservice-app -authority https://adfs.server.com/adfs
```

If AD FS server is configured properly new token should be returned to console.

Troubleshooting

To effectively trouble shoot issues with AD FS Server is best to go Event Viewer on AD FS server and select following directory: Event View (Local) > Applications and Services Logs > AD FS > AdminGroup

Most of the errors are self-explanatory, below there are couple of the most common ones.

Problems with generating token

1. Validate if Federation Service Identifier is properly set in AD FS server.
   - Open AD FS Management window (Server Manager > Tools > AD FS Management)

- In this windows click on AD FS directory and on the right pane (Actions) select "Edit Federation Service Properties"
- Verify that Federation Service Identifier ends with "/adfs". If there is anything else after "/adfs" remove it and save changes

2. Client Secret is invalid
   - Verify that client secret for specified service principal is correct
   - Sometimes AD FS server blocks old Secret Keys, follow instruction above to reset secret key

Problems with login into Ellipse APM

1. Verify that all secrets are correct in Ellipse APM configuration
2. Verify if Redirect Url is properly setup in AD FS configuration for apm-webservice-app
   - If redirect url is not correct there is always an error in AD FS Event Viewer with information about redirect url that was used this time
   - Add this redirect URL to list in AD FS configuration:
     - Open AD FS management, select folder Application Groups
     - Select $APP_GROUP_ID that was created in previous steps. Click Properties on Action pane
     - From Applications list select $APP_GROUP_ID-webservice-app and click "Edit..."
     - Add URL to the list and save all the changes

Downloading Kubernetes Command Line Interface

Kubernetes Command Line Interface is required to install Ellipse APM solution Microservices Orchestration Platform. Kubernetes CLI later mentioned as **kubectl** can be downloaded manually from Kubernetes website, or can be downloaded using script provided with Ellipse APM installation package

To download kubectl execute using provided script use following steps:

1. Download Ellipse APM installation package
2. Unzip it to directory
3. Go to **/apminstall/InstallationPackage/InstallScripts**
4. Verify that KubectlDownloader.ps1 is this directory
5. Verify that your internet connection is working properly and it's not blocked by firewall, proxy.
6. Run command: .\KubectlDownloader.ps1

7. Verify that kubectl.exe file was downloaded properly to: /apminstall/InstallationPackage/InstallScripts

HA Proxy requirements for installation on Ellipse APM

Microservices Orchestration Platform provides HA Proxy service that enables load balancing of services and also SSL (TLS) termination

Configuration of HA Proxy before APM installation is started is required to successfully create whole solution, including tenant creation. Before configuration can be started following information should be prepared:

1. Certificates that will be used by web application and feeder API
2. Fully qualified hostnames that will be used for Web Application and Feeder API. DNS should be configured in a way that both hostname should be resolved to HA Proxy machine IP.
3. Node Ports for Web Application and Feeder API. Node port kubernetes is used to provide access into kubernetes service outside of its cluster vnet. NodePort number can be random, any number from range 30000-32767 is valid. It's important that Node Port should be unique for each services that will be setup on Microservices Orchestration Platform cluster.

In case of APM at least 2 services require Node Ports (Feeder API and Web Application), both this value must differ. In case of multi environment configuration on single Microservices Orchestration Platform Cluster, each environment require separate set of ports.

List of parameters to be prepared

Parameters	Descriptions	Example value
WEB_APP_HOSTNAME	Hostname under which Web App can be found. DNS should resolve this hostname to HA Proxy server	web-ellipse-apm.somedomain.com
FEEDER_API_HOSTNAME	Hostname under which Feeder API can be found. DNS should resolve this	api-ellipse-apm.somedomain.com

	hostname to HA Proxy server	
WEB_APP_API_NODEPORT	Node port number for Web Application service on kubernetes. Any value from range 30000-32767. Must be unique across all other node ports in Microservices Orchestration Platform cluster.	30080
FEEDER_API_NODEPORT	Node port number for Feeder API service on kubernetes. Any value from range 30000-32767. Must be unique across all other node ports in Microservices Orchestration Platform cluster.	30880
ENVIRONMENT_NAME	Name of the environment that will be configured. Should be the same as the name of namespace on kubernetes. Value doesn't need to be exactly the same, but should be easily identifiable	apm-test

Data that should be provided by Microservices Orchestration Platform team after installation

Parameters	Descriptions	Example value
K8S_MASTER_HOSTNAME	Hostname of Kubernetes master node	apm-staging-

		master-001
K8s_NODE_NNN_HOSTNAME	NNN should be replaced by numeric identifier of a node. If Microservices Orchestration Platform has 2 nodes for kubernetes (not counting master node) than **NNN** are **001** and **002**. Hostnames for each node should be prepared.	apm-staging-node-001
K8s_MASTER_PRIVATE_IP	Ip address of Kubernetes master node (it should be private address)	172.20.0.7
K8s_NODE_NNN_PRIVATE_IP	NNN should be replaced by numeric identifier of a node. If Microservices Orchestration Platform has 2 nodes for kubernetes (not counting master node) than **NNN** are **001** and **002**. Ip address of Kubernetes node (it should be private address)	172.20.0.8

Having all of this configuration of feeder API should be done accordingly:

1. Login to machine with HAProxy installed (separate machine in some Microservices Orchestration Platform version, or master in other)
2. Create a backup of /etc/haproxy/haproxy.cfg: sudo cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg_bck
3. Stop **haproxy.service** sudo systemctl stop haproxy.service
4. Check configuration file content: sudo cat /etc/haproxy/haproxy.cfg

   HAProxy configuration should look alike:

```bash global log 127.0.0.1 local1 log /dev/log local1 notice chroot /var/lib/haproxy stats timeout 30s user haproxy group haproxy daemon

Defaults log global mode http option httplog option dontlognull timeout connect 5000 timeout client 50000 timeout server 50000

Frontend http_front bind *:80 stats uri /haproxy?stats default_backend http_back

Backend http_back balance roundrobin server apm-ci-cl312er-node-001 172.20.0.4:5671 check ```

1. Copy your certificates to /etc/ssl/haproxy copy *.pem /etc/ssl/haproxy/
```

2. Edit haproxy configuration file preserving existing configuration (add lines at the bottom of the configuration file)

```bash sudo vim /etc/haproxy/haproxy.cfg

Frontend https_front bind *:443 ssl crt /etc/ssl/haproxy

    acl host_<ENVIRONMENT_NAME>_api hdr_dom(host) -i <FEEDER_API_HOSTNAME>
    acl host_<ENVIRONMENT_NAME>_web hdr_dom(host) -i <WEB_APP_HOSTNAME>

    use_backend <ENVIRONMENT_NAME>_api_back if host_<ENVIRONMENT_NAME>_api
    use_backend <ENVIRONMENT_NAME>_web_back if host_<ENVIRONMENT_NAME>_web

backend _api_back balance roundrobin server  : check server  :check server  :check
#(repeat for each node)

backend _web_back balance roundrobin server  : check server  :check server  :check
#(repeat for each node)

```

1. Start haproxy service sudo systemctl start haproxy.service

2. Verify that both and are available (for example use curl)

3.

Example of proper haproxy configuration in environment with single master node and single worker node on kubernetes

```
global
  log 127.0.0.1 local1
  log /dev/log local1 notice
  chroot /var/lib/haproxy
  stats timeout 30s
  user haproxy
  group haproxy
  daemon

defaults
  log global
  mode http
  option httplog
  option dontlognull
  timeout connect 5000
```

```
  timeout client 50000
  timeout server 50000

frontend https_front
  bind *:443 ssl crt /etc/ssl/haproxy
     acl host_abb-apm_api hdr_dom(host) -i api-ci.ourhostname.abb
     acl host_abb-apm_web hdr_dom(host) -i apm-ci.ourhostname.abb

     use_backend abb-apm_api_back if host_abb-apm_api
     use_backend abb-apm_web_back if host_abb-apm_web

backend abb-apm_api_back
     balance roundrobin
     server apm-ci-master-001 172.20.0.9:30080 check
     server apm-ci-node-001 172.20.0.10:30080 check

backend abb-apm_web_back
     balance roundrobin
     server apm-ci-master-001 172.20.0.9:30880 check
     server apm-ci-node-001 172.20.0.4:100880 check
```

Troubleshooting

1. If feeder API and web app hostnames are not resolved, please verify that DNS configuration contains information about them.
2. If there is information about missing certificates, or not trusted certificates please verify if /etc/ssl/haproxy service contains proper files

Configuring Map Provider Service

Ellipse APM is using Map Provider Service in multiple of its core functionalities. Customer should provide access to one of the supported Map Services. Microservices Orchestration Platform should have access to this server.

Currently supported and certified Map Provider Services for Ellipse APM are:

- Bing maps
- OpenStreet maps

Configuration of Map provider should be stored in Customer definition JSON file. See: Customer.json explained. If there is no map provider configuration in Customer JSON file, than Ellipse APM will be using Bing Maps with key configured during setup. Default key is stored in: **DEFAULT_MAP_CONFIGURATION_KEY**. See:

Create common namespace. Below you can find examples of proper Map configuration for different map providers:

Example configurations

Bing

```
"MapProviderConfiguration": {
    "Type": "Bing",
    "MapProviderUriSchema": "", - for BING maps should be empty
    "Key": "API Key for Bing Maps" -Should be provided by Customer
  }
```

OpenStreet Maps

```
"MapProviderConfiguration": {
    "Type": "OpenStreetMaps",
    "MapProviderUriSchema": "https://[openstreetmaps-server-url]/{z}/{x}/{y}.png",
    "Key": "" - Not required for OpenStreetMaps
  }
```

Installation phases

Before installation can be executed please verify that all prerequisites are met. Ellipse APM On-Premise Prerequisites

1. Cleaning up tenant namespace on Microservices Orchestration Platform Cluster (optional)
2. Cleaning up common namespace on Microservices Orchestration Platform Cluster (optional)
3. Create common namespace
4. Create Tenant Namespace on Microservices Orchestration Platform cluster

Cleaning up tenant namespace on Microservices Orchestration Platform

Prerequisites

- Windows machine with Installation package downloaded and unzipped to ,
- PowerShell Console opened (version at least 4.0),
- Kubernetes CLI downloaded and saved in /apminstall/InstallationPackage/InstallScripts,
- Kubernetes CLI configuration file (later known as **kubeconfig**)

Procedure

1. Open PowerShell and change current directory to /apminstall/InstallationPackage/InstallScripts
2. Verify if script K8sDeleteTenantNamespace.ps1 is available
3. Prepare values for following parameters

Parameters	Descriptions
ENVIRONMENT_NAME	Name of environment which contain customer (tenant) that will be deleted (DEV, QA etc.)
CUSTOMER	abb-test
KUBECONFIG_LOCATION	C:\APM_install\mop_k8s.config

Example of parameters definition

```
$ENVIRONMENT_NAME = "abb-test"
$KUBECONFIG_LOCATION = "C:\APM_install\mop_k8s.config"
$CUSTOMER = "abb"
```

Run Command:

```
.\K8sDeleteTenantNamespace.ps1 -NamespacePrefix $ENVIRONMENT_NAME -K8sConfigPath $KUBECONFIG_LOCATION -Customers $CUSTOMER
```

To check existing environments and tenants created on Microservices Orchestration Platform please follow instructions below:

- Login to Kubernetes dashboard
- Click **Namespaces**
- Verify existing namespaces

Example List of namespaces:

- env-qa
- env-qa-abb
- default

In this example ENVIRONMENT_NAME is env-qa and NAMESPACE_NAME is abb

Cleaning up common namespace on Microservices Orchestration Platform

Prerequisites

- Windows machine with Installation package downloaded and unzipped to ,

- PowerShell Console opened (version at least 4.0),
- Kubernetes CLI downloaded and saved in /apminstall/InstallationPackage/InstallScripts,
- Kubernetes CLI configuration file (later known as **kubeconfig**)

Procedure

1. Open PowerShell and change current directory to /apminstall/InstallationPackage/InstallScripts
2. Verify if script K8sDeleteCommonNamespace.ps1 is available
3. Prepare values for following parameters

Parameters	Descriptions
ENVIRONMENT_NAME	Name of the environment to remove, eg. DEV, QA, PROD
KUBECONFIG_LOCATION	Path to **kubeconfig**. Full path should be provided

Example of parameters definition

```
$ENVIRONMENT_NAME = "abb-test"
$KUBECONFIG_LOCATION = "C:\APM_install\mop_k8s.config"
```

Run Command:

```
.\K8sDeleteCommonNamespace.ps1 -NamespacePrefix $ENVIRONMENT_NAME -K8sConfi
gPath <$KUBECONFIG_LOCATION
```

Checking environments and tenants

To check existing environments and tenants created on Microservices Orchestration Platform please follow instructions below:

- Login to Kubernetes dashboard
- Click **Namespaces**
- Verify existing namespaces

Example List of namespaces:

```
* `env-qa`
* `env-qa-abb`
* `default`
```

In this example ENVIRONMENT_NAME is env-qa

Create environment

Prerequisites

- Windows machine with Installation package downloaded and unzipped to ,
- PowerShell Console opened (version at least 4.0),
- Kubernetes CLI downloaded and saved in /apminstall/InstallationPackage/InstallScripts,
- Kubernetes CLI configuration file (later known as **kubeconfig**)
- Customer JSON file created and saved into See: Customer.json explained
- Environment configuration JSON file
- Tenant configuration JSON file

Procedure

1. Open PowerShell and change current directory to /apminstall/InstallationPackage/InstallScripts
2. Verify if script: K8sCreator.ps1 is available
3. Prepare values for following parameters

Parameters	Descriptions
ENVIRONMENTCONFIGDATAPATH	Path to json file containing configuration for common path of environment. Full path including filename should be specified.
TENANTS	List of tenants to deploy.

Example of parameters definition

```
$ ENVIRONMENTCONFIGDATAPATH = "C:\InstallationPackage\EnvironmentsConfigs\env.json "
$TENANTS = @(“Tenant1”, “Tenant2”)
```

Run Command:

Note: **Before running this command, fill configuration json files.**

```
.\K8sCreator.ps1 -EnvironmentConfigDataPath “$ENVIRONMENT_FILE” -TENANTS @(“Tenant1”, “Tenant2”)
```

Environment configuration file

Prerequisites

- Windows machine with Installation package downloaded and unzipped to ,
- Environment configuration JSON file

Procedure

1. Open directory /apminstall/InstallationPackage/InstallScripts/EnvironmentsConfigs
2. Open/create json file with the name of environment
3. Prepare values for following parameters

Parameters	Descriptions
AAD_APPLICATION_ID	Web application service principal
AAD_DIRECTORY_AUTHORITY	Authority URL
AAD_DIRECTORY_ID	Leave blank
AD_SUPPORT_CONTACT_URL	Support mail contact
APPLICATION_SECRET_KEY_BASE64	Application secret key (later is encoded to BASE64)
APPLICATIONINSIGHTSNAME	Leave blank
BUS_CONFIGURATION_BASE64	Bus configuration string (later is encoded to BASE64)
BUS_TYPE	Type of bus (e.g. kafka)
COMMONNAMESPACEPLACEHOLDERS	Leave blank
DEFAULT_MAP_CONFIGURATION_KEY_BASE64	Key for map
DOCKERPASSWORD	Docker user password
DOCKERREPOSITORYNAME	Name of docker repository
DOCKERSERVER	Docker server address
DOCKERUSERNAME	Docker user name
FEEDERAPIPORT	Leave blank
INSTALLPACKAGEROOTDIRECTORY	Directory in which all installation resources are placed
K8SCONFIGPATH	Path to **kubeconfig**
KEY_VAULT_APPLICATION_ID	Leave blank

KEY_VAULT_APPLICATION_KEY_BASE64	Leave blank
KEY_VAULT_KEY_ADDRESS	Leave blank
NAMESPACEPREFIX	Prefix for kubernetes namespace
POWER_BI_ACCESS_KEY_BASE64	Leave blank
POWER_BI_WORKSPACE_COLLECTION	Leave blank
RESOURCEGROUPLOCATION	Leave blank
RESOURCEGROUPNAME	Leave blank
SERVER	Address of web application
SQL_CONNECTIONSTRING_BASE64	Sql connection string (later is encoded to base64)
SQLSERVERLOGIN	Sql server login
SQLSERVERNAME	Sql server name
STORECONFIGURATIONSTRING	Fill with sql connection string
STORETYPE	Type of store (e.g. SqlDatabase)
WEBSERVICEPORT	Port on which web application is present
WEBSITENAME	Name of web application

Example of variables definitions:

```
APPLICATIONINSIGHTSNAME = ""
AAD_APPLICATION_ID = "apm-webservice-app"
AAD_DIRECTORY_AUTHORITY = "https://qa-adfs.ourhostname.com/adfs"
AAD_DIRECTORY_ID = ""
AD_SUPPORT_CONTACT_URL = https://ad.provider/services/maintenance/support
APPLICATION_SECRET_KEY_BASE64 = "sadawer123"
BUS_CONFIGURATION_BASE64 = "Brokers=127.0.0.1:9092;"
BUS_TYPE = "kafka"
KEY_VAULT_APPLICATION_ID = ""
KEY_VAULT_APPLICATION_KEY_BASE64= ""
KEY_VAULT_KEY_ADDRESS = ""
POWER_BI_ACCESS_KEY_BASE64 = ""
POWER_BI_WORKSPACE_COLLECTION = ""
DEFAULT_MAP_CONFIGURATION_KEY_BASE64 = ""
DOCKERPASSWORD = "Dockerpasword1 "
DOCKERREPOSITORYNAME = "app-repository"
DOCKERSERVER = "ahck8s.azurecr.io"
```

```
DOCKERUSERNAME = "ahc_user"
FEEDERAPIPORT = "8000"
INSTALLPACKAGEROOTDIRECTORY = "C:\InstallationPackage"
K8SCONFIGPATH = "C:\APM_install\mop_k8s.config"
NAMESPACEPREFIX = "namespace-prefix"
RESOURCEGROUPLOCATION =" "
RESOURCEGROUPNAME = ""
SERVER = " https://ellipse-apm-qa.somedomain.com"
SQL_CONNECTIONSTRING_BASE64 = "Data Source=172.0.0.11,50234;Initial Catalog=AHC-T
ENANT;User ID=sa;Password=DifficultPassword1; "
SQLSERVERLOGIN ="ahc_user"
SQLSERVERNAME = "sql-server-name"
STORECONFIGURATIONSTRING = “Data Source=172.0.0.11,50234;Initial Catalog=AHC-TEN
ANT;User ID=sa;Password=DifficultPassword1;"
STORETYPE = " SqlDatabase "
WEBSERVICEPORT = "8080"
WEBSITENAME = "ellipse-apm-qa"
```

Exemplary file:

```
{
    "ApplicationInsightsName": "",
    "CommonConfigPlaceholders": {
        "<AAD_APPLICATION_ID>": "apm-webservice-app",
        "<AAD_DIRECTORY_AUTHORITY>": " https://qa-adfs.ourhostname.com/adfs ",
        "<AAD_DIRECTORY_ID>": "",
        "<AD_SUPPORT_CONTACT_URL>": " https://ad.provider/services/maintenance/support",
        "<BUS_TYPE>": "kafka",
        "<KEY_VAULT_APPLICATION_ID>": "",
        "<KEY_VAULT_KEY_ADDRESS>": "",
        "<POWER_BI_WORKSPACE_COLLECTION>": ""
    },
    "CommonNamespacePlaceholders": {},
    "CommonSecretsPlaceholders": {
        "<APPLICATION_SECRET_KEY_BASE64>": "sadawer123",
        "<BUS_CONFIGURATION_BASE64>": "Brokers=127.0.0.1:9092;",
        "<KEY_VAULT_APPLICATION_KEY_BASE64>": "",
        "<SQL_CONNECTIONSTRING_BASE64>": "Data Source=172.0.0.11,50234;Initial Catalog=AHC-TENANT;User
ID=sa;Password=DifficultPassword1;",
        "<POWER_BI_ACCESS_KEY_BASE64>": "",
        "<WINDOWS_ACTIVE_DIRECTORY_USER_PASSWORD_BASE64>": "",
        "<DEFAULT_MAP_CONFIGURATION_KEY_BASE64>": ""
    },
    "DockerPassword": "Dockerpasword1",
    "DockerRepositoryName": "app-repository",
    "DockerServer": "ahck8s.azurecr.io",
    "DockerUserName": "ahc_user",
    "FeederApiPort": "",
    "InstallPackageRootDirectory": "C:\InstallationPackage",
    "K8sConfigPath": "C:\APM_install\mop_k8s.config",
    "NamespacePrefix": "namespace-prefix",
    "ResourceGroupLocation": "",
    "ResourceGroupName": "",
    "Server": "https://ellipse-apm-qa.somedomain.com",
    "SqlServerLogin": "ahc_user",
    "SqlServerName": "sql-server-name",
    "StoreConfigurationString": "Data Source=172.0.0.11,50234;Initial Catalog=AHC-TENANT;User
ID=sa;Password=DifficultPassword1;",
    "StoreType": "SqlDatabase",
```

```
    "WebServicePort": "8080",
    "WebSiteName": "ellipse-apm-qa"
}
```

Tenant configuration file

Prerequisites

- Windows machine with Installation package downloaded and unzipped to ,
- Environment configuration JSON file

Procedure

1. Open directory /apminstall/InstallationPackage/InstallScripts/EnvironmentsConfigs
2. Open/create json file with the name of environment
3. Prepare values for following parameters

Parameters	Descriptions
APPLICATION_SECRET_KEY_BASE64	Application secret key (later is encoded to BASE64)
BUSCONFIGURATIONSTRING	Bus configuration string
FEEDER_CLIENT_SECRET_BASE64	Web application service principal
NOTIFICATION_ALERT_EMAIL_ADDRESS	Address of notification email
NOTIFICATION_EMAIL_SERVER	Notification email server
NOTIFICATION_EMAIL_SERVER_ENABLE_SSL	Enable SSL for notification email
NOTIFICATION_EMAIL_SERVER_PASSWORD	Notification email username
NOTIFICATION_EMAIL_SERVER_PORT	Notification email server port
NOTIFICATION_EMAIL_SERVER_USERNAME	Notification email username
SQLCONNECTIONSTRING	Sql connection string
STORECONFIGURATIONSTRING	Store configuration string
SUPERADMINCLIENTID	Super admin client id
SUPERADMINCLIENTSECRET	Super admin password

Example of variables definitions:

```
APPLICATION_SECRET_KEY_BASE64 = "apm-webservice-app"
BUSCONFIGURATIONSTRING = "Brokers=127.0.0.1:9092;"
FEEDER_CLIENT_SECRET_BASE64 = "apm-webservice-app"
NOTIFICATION_ALERT_EMAIL_ADDRESS = "alert@ourserver.com"
NOTIFICATION_EMAIL_SERVER = "smtp.atsomeserver.net"
NOTIFICATION_EMAIL_SERVER_ENABLE_SSL = "true"
```

```
NOTIFICATION_EMAIL_SERVER_PORT = "2525"
NOTIFICATION_EMAIL_SERVER_USERNAME = "apikey"
NOTIFICATION_EMAIL_SERVER_PASSWORD = "EmAiLServERPasWORD"
SQLCONNECTIONSTRING = " Data Source=172.0.0.11,50234;Initial Catalog=AHC-TENANT;U
ser ID=sa;Password=DifficultPassword1;"
STORECONFIGURATIONSTRING = " Data Source=172.0.0.11,50234;Initial Catalog=AHC-TEN
ANT;User ID=sa;Password=DifficultPassword1;"
SUPERADMINCLIENTID = "apm-webservice-superadmin"
SUPERADMINCLIENTSECRET = "SeCrETFoRSuPerAdMiN"
```

Exemplary file:

```
{
    "BusConfigurationString": "Brokers=127.0.0.1:9092;",
    "TenantConfigurationPlaceholders": {
        "<NOTIFICATION_ALERT_EMAIL_ADDRESS>": "alert@ourserver.com",
        "<NOTIFICATION_EMAIL_SERVER>": "smtp.atsomeserver.net",
        "<NOTIFICATION_EMAIL_SERVER_ENABLE_SSL>": "true",
        "<NOTIFICATION_EMAIL_SERVER_PORT>": "2525",
        "<NOTIFICATION_EMAIL_SERVER_USERNAME>": "apikey"
    },
    "TenantNamespacePlaceholders": {},
    "TenantSecretsPlaceholders": {
        "<APPLICATION_SECRET_KEY_BASE64>": "sadawer123",
        "<FEEDER_CLIENT_SECRET_BASE64>": "apm-webservice-app",
        "<NOTIFICATION_EMAIL_SERVER_PASSWORD>": "EmAiLServERPasWORD"
    },
    "SQLConnectionString": "Data Source=172.0.0.11,50234;Initial Catalog=AHC-TENANT;User
ID=sa;Password=DifficultPassword1;",
    "StoreConfigurationString": "Data Source=172.0.0.11,50234;Initial Catalog=AHC-TENANT;User
ID=sa;Password=DifficultPassword1;",
    "SuperAdminClientId": "apm-webservice-superadmin",
    "SuperAdminClientSecret": "SeCrETFoRSuPerAdMiN"
}
```

Tenant

Cloud based APM is wannabe multi-tenant application. Multi-tenancy is an architecture in which a single instance of a software application serves multiple customers. Each customer is called a tenant. We can create new Tenants by using Customer.ps1 PowershellScript.

Prerequisites:

-Customer.ps1 script – ask Ellipse APM DevOps team for latest version of the script.

-SuperAdminClientId and SuperAdminClientSecret

-Identity provider (Azure Active Directory) identifier.

-Prepare script similar

-Copy it to Customers folder.

Customer.json

- {Customer}.json file holds basic information about Tenant and it will be loaded via Customer.ps1 script and used as reference in case user would want to later update the configuration.
- Example file:

Example Customer JSON file

{


```
{
 "InternalName": "ABB",
 "CustomerName": "ABB",
 "CustomerAADCredentials": [
  {
   "IdentityProvider": "https://adfs.server.com/adfs",
   "SecurityGroupGuid": null
  }
 ],
 "PowerBiDatabasePassword": "************************************",
 "EllipseServiceConfig": {
  "WebServicesAddress": "http://eam.server.com",
  "UIAddress": "http://eam.server.com/html/ui",
  "TenantId": "ffd98dfb-aaaa-bbbb-cccc-5f1433e04bef",
  "ApplicationClientId": "fb2eb9d6-0e34-4fc2-9ada-ggggggggggg",
  "InstanceTimeZoneName": "US Mountain Standard Time",
  "Position": "SYSAD",
  "DistrictCode": "0000",
  "UserName": "USER_NAME@EAM",
  "Password": "************************",
  "Classification": "RI",
  "WorkGroup": "",
  "InspectionServiceBusEndpoint": "",
  "InspectionServiceBusQueueName": ""
 },
 "MapProviderConfiguration": {
```

```
    "Type": "MapType",
    "MapProviderUriSchema": "MapProviderServerUrl",
    "Key": "ApiKey"
  },
  "LdapConfiguration": {
    "LdapServerAddress": "ad.server.com",
    "LdapServerPort": "389",
    "UseSecureSocketLayer": false,
    "LdapFetchDataUserLogin": "aduser",
    "LdapFetchDataUserPassword": "AdUserPassword",
    "LdapTlsCertificate": null,
    "WindowsActiveDirectoryRolesJsonDictionary": "{\"CN=EngineersAPM,CN=Users,DC=qa-adfs,DC=dev,DC=apm,DC=enterprisesoftware,DC=abb\":\"Engineer\",\"CN=AdministratorApm,CN=Users,DC=qa-adfs,DC=dev,DC=apm,DC=enterprisesoftware,DC=abb\":\"Administrator\"}"
  }
}
```


- Customer Configuration Reference
- InternalName – customer (tenant) internal name, used for example as suffix of SQL Database.
- CustomerName – customer name, displayed on UI.
- CUSTOMERAADCREDENTIALS – dodać wzmianke, że zawiera te pary identityprovider i security group
- **IdentityProvider** – customer's Azure Active Directory, only users from this AAD will see customers data (see Identity provider for more details).
- **SecurityGroupGuid** - not required for onPrem
- PowerBiDatabasePassword – SQL user password which has read-only access to SQL Database (see SQL user credentials for more details).
- EllipseServiceConfig – configuration of Ellipse integration, e.g. URL and credentials (see How to configure Ellipse integration? for more details).
- MapProviderConfiguration - configuration of Map Provider used in Ellipse APM. See: Configuring Map Provider Service
- LdapConfiguration - Configuring Active Directory Management user

Customer.ps1 script

- Script provides methods to create/update customer and populate standard PowerBI reports.
- Arguments:

  -Server
  URL to the Ellipse APM server

Mandatory: yes
-Authority
ID of the AAD directory/tenant
Mandatory: yes
-AadApplicationId
ID of the AAD application registered with a tenant
Mandatory: yes
-SuperAdminClientId
Client ID of service principal with super-admin role
Mandatory: yes
-SuperAdminClientSecret
Client secret of service principal with super-admin role
Mandatory: yes
-Customer
Customer file name with customer configuration
Mandatory: yes
-Create
Creates customer based on configuration file. If customer already exist then Update i s applied.
Mandatory: no
-Update
Updates customer based on configuration file
Mandatory: no
-CreateNoUpdate
Creates customer if not exists.
Mandatory: no
-PowerBIs
Populates standard reports
Mandatory: no

Create Tenant

To create XYZ customer create XYZ.json file copy it to Customer folder and run script:

```
.\Customer.ps1 `
-Server "server" `
-Customer "XYZ" `
-Authority "tenant id" `
-AadApplicationId "resource application id here" `
-SuperAdminClientId "super admin client id" `
```

-SuperAdminClientSecret "super admin secret" `
-Create

Creation of principal role

Each Ellipse APM environment has its SuperAdmin Id and Secret that is needed to create new customer (tenant). If you don't know your SuperAdmin credentials please contact Ellipse APM team.

Execute Once .json is filled and copied to Customers folder you can execute the script:

.\Customer.ps1 -Server "server" -Authority "tenantId" -AadApplicationId "resourceAppId" -SuperAdminClientId "superAdminClientId" -SuperAdminClientSecret "superAdminCientSecret" -customer "customer" -Create -PowerBIs

Where: * server is URL of Ellipse APM environment, * login and key are described in Super-admin login and key, * Customer is a Customer configuration file name (without .json).

Notice that in this case client id and client secret for SuperAdmin should be provided.

Back to main page ## Full Update - Updating customer configuration is possible by redefining Customers.json file and calling Customer.ps1 script – It must be invoked as SuperAdmin account. Customers.json format is defined in "Customer.json explained" section. To update XYZ customer modify properly XYZ.json file and run Customer.ps1 script with -update argument

.\Customer.ps1
-Server "server"
-Authority "tenantId"
-AadApplicationId "resourceAppId"
-SuperAdminClientId "superAdminClientId"
-SuperAdminClientSecret "superAdminCientSecret"
-customer "customer"
-Update

Partial Update by Customer

- Customer has access to his configuration and he is able to modify his secrets and configurations for integrated systems like OSI PI or Ellipse

API to Update Customer Configuration

Ellipse

URL: /api/customer/ellipse
Method: PUT
Body:


```
{
   "WebServicesAddress": "http://demo.co",
   "UIAddress": "http://demo.co/ui.html",
   "TenantId": "…",
   "ApplicationClienId": "…",
   "InstanceTimeZoneName": "…",
   "Position": "",
   "DistrictCode": "R400",
   "UserName": "username",
   "Password": "password",
   "Classification": "A1",
   "WorkGroup": "",
   "InspectionServiceBusEndpoint": "…",
  "InspectionServiceBusQueueName": "…",
}
```


Customer Configuration Reference

- InternalName – customer (tenant) internal name, used for example as suffix of SQL Database.
- CustomerName – customer name, displayed on UI.
- IdentityProvider – customer's Azure Active Directory, only users from this AAD will see customers data (see Identity provider for more details).
- SecurityGroupGuid* - needed when creating new tenant with ABB users, security group shall be created.
- PowerBiDatabasePassword – SQL user password which has read-only access to SQL Database (see SQL user credentials for more details).
- EllipseServiceConfig – configuration of Ellipse integration, e.g. URL and credentials (see How to configure Ellipse integration? for more details).

Other APM Functionalities

Intentionally blank page

Custom Translations Setup

Adding Custom translations is possible by redefining translations.{locale}.json file and calling PopulateData.ps1 script.

Form of translations file is as follows:

```
{
"translation.id": "language specific text",
"another.translation.id": "another text"
}
```

Where: {locale} is language code, for which translation will be used, e.g. translations.en.json.

Custom Translations collateral use

Besides the translation, Custom Translations can be used to provide friendly names for:

- degradation sub-scores
- model parameters
- asset sub-types

e.g.:

```
{
"Kinectrics.SF6CircuitBreaker.subscores.leakage": "Leakage",
"ModelParameter.Kinectrics.SF6CircuitBreaker.Moisture": "Moisture",
"ABB.TransformerStandard.asset.nameplate.assetsubtype.phase_shifter": "Phase Shifter"
}
```

Setting of an Authentication for Restful API

Ellipse APM has Restful API for importing all kinds of data into a system. To properly send inputs user or application has to use confidential service principal with "Import" role generated during ADFS setup.

Acquiring Authentication Token

Installation package contains PowerShell script for retrieving authentication token from application authority.

GetAuthenticationToken script usage

1. Prepare following parameters

Parameter Name	Description	Example Value
CLIENT_ID	ID of service principal with "Import" role	apm-webservice-import
CLIENT_KEY	Secret Key generated for service principal with "Import" role	2v6X66A9JGbYauPV15AcAXhbDHl7rPPKLEey-Xx4
RESOURCE_APP_ID	ID of APM application service principal	apm-webservice-app
AUTHORITY	URL of identity provider service	https://company.com/adfs

2. Run script

   .\GetAuthenticationToken.ps1 -clientId $CLIENT_ID -clientKey $CLIENT_KEY -resourceAppId $RESOURCE_APP_ID -authority $AUTHORITY

Getting token using Microsoft.IdentityModel.Clients.ActiveDirectory library

This applies to applications written in .NET and .NET Core.

1. Install newest nuget package

- Package Manager:

  Install-Package Microsoft.IdentityModel.Clients.ActiveDirectory

- .Net CLI:

  dotnet add package Microsoft.IdentityModel.Clients.ActiveDirectory

2. Use Microsoft.IdentityModel.Clients.ActiveDirectory namespace:

C# using Microsoft.IdentityModel.Clients.ActiveDirectory;

3. Create ClientCredential:

C# var clientCredential = new ClientCredential(CLIENT_ID, CLIENT_KEY);

4. Create AuthenticationContext:

C# var authenticationContext = new AuthenticationContext(AUTHORITY, false);

5. Acquire token:

C# var token = await authenticationContext.AcquireTokenAsync(RESOURCE_APP_ID, clientCredentials)

Preparing html request to API

1. Http request method: POST
2. Required headers:
- Content-Type: Application/json
- Authorization: Bearer <token>
3. Body format(just example, messages may differ based on type):

```
{
  "messages":[
    {
      "type": "type of data",
      "parameterName": "temperature",
      "value": 26.5,
      "timestamp": "2018-01-31T08:45:00.000Z"
    }
  ]
}
```

Monitoring Tools

On Premise version of Ellipse APM is utilizing monitoring and logging mechanism provided by Microservices Orchestration Platform. Ellipse APM is not providing any additional tools, so roles required for monitoring and logging mechanism should be included into Microservices Orchestration Platform cluster creation.

To monitor environment status and gather logs from whole environment each node on Microservices Orchestration Platform should include following roles:

- filebeat
- node_exporter

To Monitor Kafka and all topics used by Ellipse APM each Kafka node should include following role:

- kafka-exporter
- jmx-exporter

Additionally at least one node should be including roles:

- grafana
- prometheus
- elasticsearch
- kibana
- elasticsearch-curator

This setup is a minimal one, however recommended set up includes two different nodes, one for monitoring and one for logging services. In recommended setup there should be two nodes:

1. Monitoring node including roles:
   - grafana
   - prometheus
2. Logging node including roles:
   - elasticsearch
   - elasticsearch-curator
   - kibana

Grafana

Grafana overview

Monitoring for On Premise solution is based on Grafana Software. Grafana is widely known, highly configurable and scalable. Grafana by default requires a source of data, and for that Prometheus is used. Prometheus is also widely known solution, with multiple plugins for different data sources. By default Ellipse APM solution is using node_exporter plugin which returns full information about the OS and Kubernetes status. Kafka_exporter is a plugin that gives more insight about status of Kafka topics, consumer groups, offsets and others.

Fresh installation of the Microservice Orchestration Platform provides fresh Grafana configured to talk with Prometheus. However there is no Grafana dashboards configured by default, so client need to include them manually (default dashboards for monitoring Ellipse APM environment is included into installation package).

Grafana default configuration

Grafana Service by default is working on port **3000**. To access Grafana dashboard open a browser and type https://grafana_host:3000/

In case of any issues with opening Grafana please ensure that:

- Certificates provided by Microservices Orchestration Platform are either properly signed, or in case of the testing environment self-signed certificates are accepted
- Port 3000 for grafana_host should be available from machine on which you opened a browser

After first login you will be asked for a credentials (*default* are admin/admin). Please change default password as soon as possible. Currently **Microservices Orchestration Platform** doesn't not support integration with Active Directories.

After successful login an empty Grafana dashboard should be visible. On this panel new users can be created, or additional plugins can be installed. This is however not required by Ellipse APM since Prometheus plugin is already preinstalled with.

Adding new dashboards

While on main page.

1. Hover over '+' icon on the left side pane.
2. Click "Import Dashboard"
3. Click "Upload .json File" button
4. Go to
5. Select one of the json files containing dashboard configurations
6. New page should be opened containing couple of the configuration options
   - Default name can be changed to anything else
   - By default Folder is General, but any other can be selected
   - Lastly from Prometheus dropdown option "Prometheus" need to be selected
7. With all of the above properly selected click "Import" button
8. New dashboard should be opened containing all kind of the metrics.

Kibana

Kibana overview

Kibana is a logging analysis tools that is currently used by Ellipse APM inside Microservices Orchestration Platform. Kibana is part of the Elastic Stack which currently is using Kibana for log presentations, Elasticserach for storing and indexing logs and Filebeat for gathering logs from multiple different parts of Microservices Orchestration Platform and Ellipse APM.

Kibana is using **lucene** query syntax to query Ellipse APM.

Kibana default configuration

Kibana service by default is working on port **5601**. To access Kibana dashboard open a browser and type https://kibana_host:5601/app/kibana

In case of any issues with opening Kibana please ensure that:

- Certificates provided by Microservices Orchestration Platform are either properly signed, or in case of the testing environment self-signed certificates are accepted
- Port 5601 for kibana_host should be available from machine on which you opened a browser

First time opening kibana new index patterns should be setup to gather logs from system.

1. Click "Management" button on the left side
2. Click "Index Patterns"
3. Wizard page for index creation should be displayed with the list of available indices
4. Indices by default should have name matches **filebeat-#version#-#date#**
5. In **Index Pattern** put filebeat-* and click "> Next Step" button
6. Select "@timestamp" from the dropdown.
7. Click "Create Index Patthern button"
8. New index should be created.

After this logs can be browsed by clicking "Discover" button. Query panel above is using Lucene syntax for logs analysis.

The end