

GELARM

Organize
Control
GIMS your data

GIMS Automation

Решение для автоматизации процедур ETL

GELARM – российский вендор, создающий системы мониторинга, инвентаризации и обработки данных (ETL), управления ИТ инфраструктурой и технологическими процессами предприятий.

Флагманским решением компании является линейка комплексных программных продуктов
Gelarm Infrastructure Management System (GIMS):

GIMS Automation

система для интеграции различных информационных систем между собой. Система обеспечивает подключения к различным источникам данных, сценарии обработки информации, триггеры для запуска сценариев обработки информации.

GIMS Monitoring

система мониторинга сервисов организации. Система обеспечивает сбор и корреляцию событий сервисов, сбор метрик производительности и оценку качества сервисов.

GIMS Inventory

система инвентаризации оборудования, программного обеспечения и объектов инфраструктуры. Система обеспечивает построение топологий инфраструктуры, создание моделей данных описывающих классы объектов, связей и их атрибуты.

5 лет
НА РЫНКЕ

20+
РАЗРАБОТЧИКОВ
А-КЛАССА

40
ПРОФЕССИОНАЛЬНЫХ
СЕРТИФИКАТОВ IBM

25+
МАСШТАБНЫХ
ПРОЕКТОВ

Особенности Gelarm Infrastructure Management System (GIMS)

- В состав комплексного решения GIMS входят полностью и бесшовно интегрируемые системы (GIMS Inventory, GIMS Monitoring, GIMS Automation)
- Открытый код, гибкий API и возможность доработки решений «под задачу»
- Нарращивание производительности происходит в 2 клика
- Единая кластерная инфраструктура всех решений и гибкое управление кластерами
- Удобный и интуитивно понятный интерфейс
- Полномасштабный портал для пользователей
- Прозрачная ценовая политика, удобная схема лицензирования и соответствие законодательству РФ
- Комбинируя компоненты тем или иным образом можно создавать различные решения, например, для инвентаризации инфраструктуры, сбора и обработки данных для наполнения КХД (ETL), работы с НСИ, построения зонтичной системы мониторинга
- Построив одно решение можно легко превратить его в другое, просто добавив необходимые компоненты (например, решение для инвентаризации может расширено до полноценной системы мониторинга инфраструктуры путем добавления компонента GIMS Monitoring)

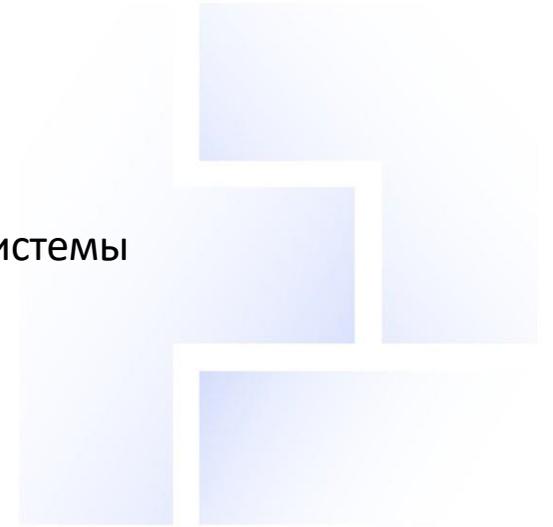
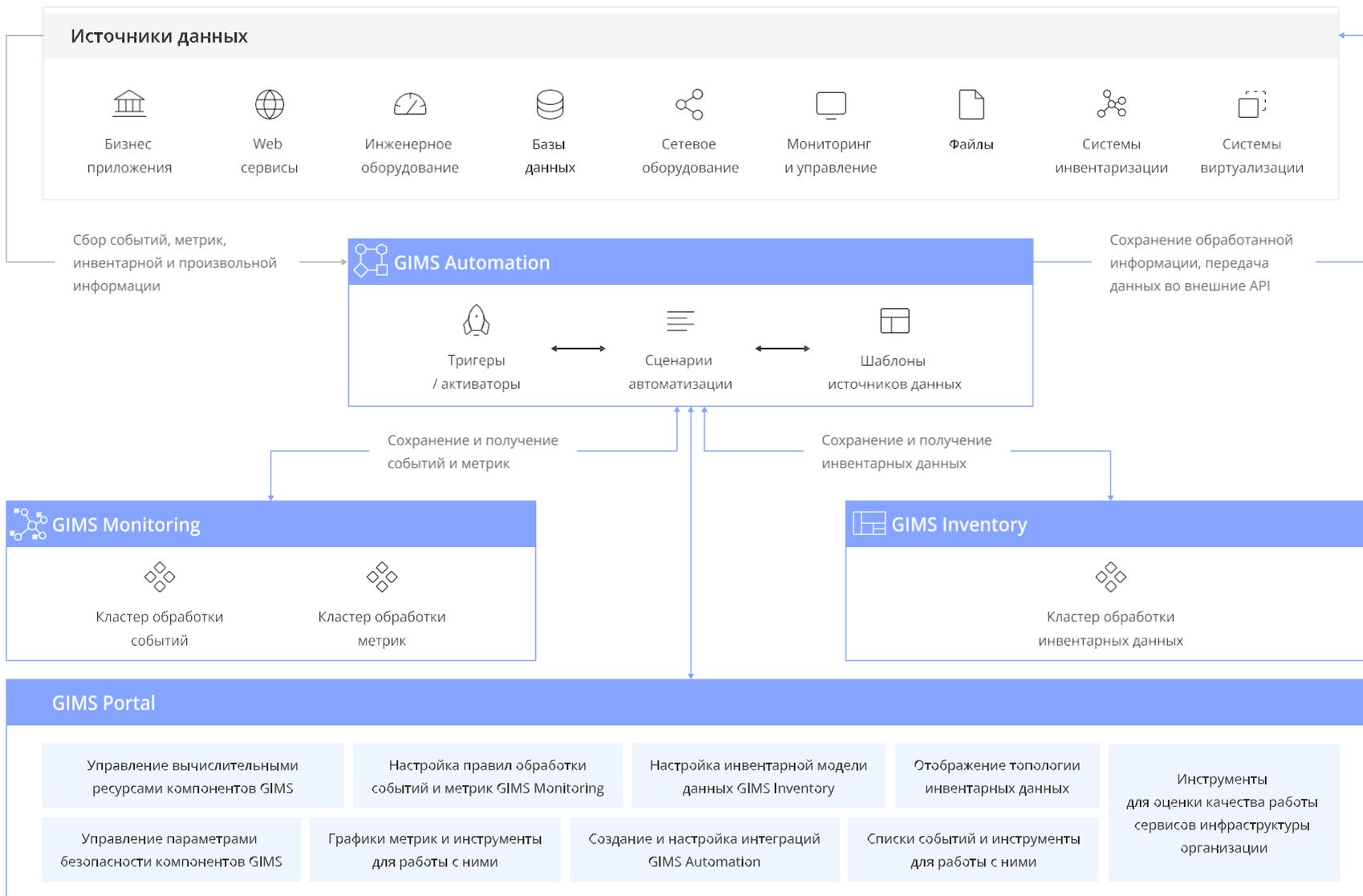


Схема взаимодействия решений, входящих в Gelarm Infrastructure Management System (GIMS)



Система обеспечивает:

- Выгрузку, обработку и выдачу данных как из, так и в различные источники, реализуя таким образом автоматизацию процедур ETL для любых СУБД и консолидацию данных из любых источников в кратчайшие сроки
- Интеграцию различных приложений между собой, реализуя функции шины данных
- Замену множества различных интеграционных решений, существенно снижая расходы на сопровождение
- Создание сценариев опроса и тестирования для различных объектов и сценариев автоматизации процессов обработки и передачи данных

Источники данных

Добавить Удалить

Имя	Тип	Статус
adb	PostgreSQL	🟢
automation01_postgres	GIMS FaultDB	🟢
case_4	HTTP	🟢
CASE_CSV	File	🟢
CASE_HIVE	Hive	🟢
csv_test_61	File_CSV	🟢
db2_test	DB2	🟢
dpolovinkin_automation	GIMS FaultDB	🟢
ESXL_Gelarm	VMWare ESX	🟢
FaultDB	GIMS FaultDB	🟢
FaultDB_LM	GIMS FaultDB	🟢
GIMS_Inventory_92	GIMS Inventory	🟢
GIMS_PerformanceDB	GIMS PerformanceDB	🟢
hive_test	Hive	🟢
inventory2	GIMS Inventory 2	🟢
mds_sandbox_pgsq1	PostgreSQL	🟢
monitor_81	GIMS FaultDB	🟢
mssql_test	MSSQL	🟢
mysql_test	MySQL	🟢
nko		🟢

automation01_postgres x

Тип источника данных

Тип источника данных: GIMS FaultDB

Общие

db_name: monitor

Аутентификация

username: monitor

password:

Сетевые настройки

port: 5432

hostname: 192.168.88.61

Сценарии автоматизации

Discovery_test x

```

1 import os, re
2 import argparse
3 import threading, concurrent.futures, queue
4 import json
5 from collections import namedtuple
6 from ipaddress import ip_network, ip_address
7 from multiprocessing.pool import ThreadPool
8 import multiprocessing
9 from os import path, chdir
10 from socket import gethostname, gethostbyaddr, getaddrinfo, gaierror
11 from subprocess import check_output, CalledProcessError
12 from sys import stdout
13 from time import time
14 from pymon.hapi import *
15 from pymon.entity.rfc3413.oneliner import oneliner
16 import subprocess
17 import ipaddress
18 import json
19 from collections import defaultdict
20
21
22 def snmpgetEntInstList():
23     # entityClassMapping = (1: 'other', 2: 'unknown', 3: 'chassis', 4: 'backplane', 5: 'container', 6: 'powerSupply', 7: 'fan', 8: 'sensor', 9: 'module', 10: 'port', 11: 'stack', 12: 'cpu')
24     components=defaultdict(dict)
25
26
27     snmp_iter = bulkCmd(SnmpEngine(),
28                       community='public',
29                       udpTransportTarget(ip, 161),
30                       contextData(),
31                       0, 50,
32                       ObjectType(ObjectIdentity('INTI-MIB', 'entPhysicalDescr').addMibSource(MibPath)),
33                       ObjectType(ObjectIdentity('INTI-MIB', 'entPhysicalContainedIn').addMibSource(MibPath)),
34                       ObjectType(ObjectIdentity('INTI-MIB', 'entPhysicalClass').addMibSource(MibPath)),
35                       ObjectType(ObjectIdentity('INTI-MIB', 'entPhysicalName').addMibSource(MibPath)),
36                       ObjectType(ObjectIdentity('INTI-MIB', 'entPhysicalSerialNum').addMibSource(MibPath)),
37                       ObjectType(ObjectIdentity('INTI-MIB', 'entPhysicalWoodName').addMibSource(MibPath)),
38                       ObjectType(ObjectIdentity('INTI-MIB', 'entPhysicalStatus').addMibSource(MibPath)),
39                       lexiconGraphMode=False)
40
41     for errorIndication, errorStatus, errorIndex, varBinds in snmp_iter:
42         # Check for errors and print out results
43         if errorIndication:
44             print(errorIndication)
45         elif errorStatus:
46             print('%s at %s' % (errorStatus.prettyPrint(),
47                               errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
48         else:
49             for (errorIndex, varBinds) in enumerate(varBinds):
50                 print('%s: %s' % (varBinds[0].prettyPrint(), varBinds[1].prettyPrint()))
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

GELARM | В чем уникальность GIMS Automation?

- Продукт предназначен для интеграции с различными объектами для решения задач инвентаризации и мониторинга, получения информации от систем управления, различных информационных систем, систем плановых работ, порталов, шин данных
- Может сам выступать в качестве шины данных для передачи информации между различными системами, в том числе, расположенных в разных контурах безопасности
- Предоставляет возможность создания консолидированной отчетности по любым процессам и/или бизнес задачам
- Поддерживает возможность создания множества кластеров для распределённой интеграции с инфраструктурой. Каждый кластер может включать в себя до 64 серверов с возможностью асинхронной обработки данных
- Установка и настройка кластеров осуществляется из единого WEB интерфейса управления
- Является заменой шины IBM Tivoli Impact, что обеспечивает интеграцию с другими продуктами линейки IBM Tivoli NetCool



- Обеспечивает подключение к системе различных источников данных: базы данных, файлы, REST, CLI, SMP, CORBA, RSS и другие
- Обеспечивает управление всеми подключёнными источниками данных из единого web интерфейса

Источники данных

[Добавить](#) [Удалить](#)

Имя	Тип	Статус
adb	PostgreSQL	🟢
automation01_postgres	GIMS FaultDB	🟢
case_4	HTTP	🟢
CASE_CSV	File	🟢
CASE_HIVE	Hive	🟢
csv_test_61	File_CSV	🟢
db2_test	DB2	🟢
dpolovinkin_automation	GIMS FaultDB	🟢
ESXI_Gelarm	VMWare ESX	🟢
FaultDB	GIMS FaultDB	🟢
FaultDB_LM	GIMS FaultDB	🟢
GIMS_Inventory_92	GIMS Inventory	🟢
GIMS_PerformanceDB	GIMS PerformanceDB	🟢
hive_test	Hive	🟢
inventory2	GIMS Inventory 2	🟢
mds_sandbox_psgsql	PostgreSQL	🟢
monitor_81	GIMS FaultDB	🟢
mssql_test	MSSQL	🟢
mysql_test	MySQL	🟢
nko	File	🟢

automation01_postgres x

Тип источника данных

Тип источника данных: GIMS FaultDB

Общие

db_name: monitor

Аутентификация

username: monitor

password:

Сетевые настройки

port: 5432

hostname: 192.168.88.61



- Инструментарий GIMS Automation позволяет создавать собственные шаблоны источников данных для подключения специфических систем
- Создание пользовательских шаблонов источников данных осуществляется в web редакторе и не требует существенных усилий

Типы источников данных

Добавить Удалить PostgreSQL x

Описание Свойства Методы

Добавить Удалить

<input type="checkbox"/>	Имя (eng) *	Описание	Тип *	Значение по умолчанию	Раздел *	Обязательный	Скрытый
<input type="checkbox"/>	db_name	Имя базы	Строковое	Значение по умолчанию	Общие	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	hostname	Имя сервера	Строковое	Значение по умолчанию	Общие	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	port	Порт	Целое	5432	Общие	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	login	Имя пользователя	Строковое	Значение по умолчанию	Аутентификация	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	password	Пароль	Строковое	*****	Аутентификация	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



- При необходимости преобразовать, изменить или выполнить любые действия с данными можно использовать сценарии автоматизации
- Создание сценариев автоматизации также осуществляется в web редакторе
- Поддерживаются любые функции и библиотеки Python, что позволяет встраивать в сценарии различные инструменты из области искусственного интеллекта (например, использовать нейросети для поиска зависимостей в обрабатываемой информации)

Сценарии автоматизации

```
Discovery_test x
1 import os, re
2 import argparse
3 import threading, concurrent.futures, queue
4 import json
5 from collections import namedtuple
6 from ipaddress import ip_network, ip_address
7 from multiprocessing.pool import ThreadPool
8 import multiprocessing
9 from os import path, chmod
10 from socket import gethostname, gethostbyname, error, gaierror
11 from subprocess import check_output, CalledProcessError
12 from sys import stdout
13 from time import time
14 from pysnmp.hlapi import *
15 from pysnmp.entity.rfc3413.oneliner import cmdgen
16 import subprocess
17 import ipaddress
18 import json
19 from collections import defaultdict
20
21
22
23 def snmpGetEntities(ip):
24     # entityClassMapping = {1: 'other', 2: 'unknown', 3: 'chassis', 4: 'backplane', 5: 'container', 6: 'powerSupply', 7: 'fan', 8: 'sensor', 9: 'module', 10: 'port', 11:
25     'stack', 12: 'cpu'}
26     components=defaultdict(dict)
27
28     snmp_iter = bulkCmd(SnmpEngine(),
29                        CommunityData('public'),
30                        UdpTransportTarget((ip, 161)),
31                        ContextData(),
32                        0, 50,
33                        ObjectType(ObjectIdentity('ENTITY-MIB', 'entPhysicalDescr')).addMibSource(MibsPath()),
34                        ObjectType(ObjectIdentity('ENTITY-MIB', 'entPhysicalContainedIn')).addMibSource(MibsPath()),
35                        ObjectType(ObjectIdentity('ENTITY-MIB', 'entPhysicalClass')).addMibSource(MibsPath()),
36                        ObjectType(ObjectIdentity('ENTITY-MIB', 'entPhysicalName')).addMibSource(MibsPath()),
37                        ObjectType(ObjectIdentity('ENTITY-MIB', 'entPhysicalSerialNum')).addMibSource(MibsPath()),
38                        ObjectType(ObjectIdentity('ENTITY-MIB', 'entPhysicalModeName')).addMibSource(MibsPath()),
39                        ObjectType(ObjectIdentity('ENTITY-MIB', 'entPhysicalFRU')).addMibSource(MibsPath()),
40                        lexicographicMode=False)
41     for errorIndication, errorStatus, errorIndex, varBinds in snmp_iter:
42         # Check for errors and print out results
43         if errorIndication:
44             print(errorIndication)
45         elif errorStatus:
46             print('%s at %s' % (errorStatus.prettyPrint(),
47                               errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
```



- Возможность создания в системе различных триггеров для запуска сценариев автоматизации: REST, SNMP traps, schedule, syslog, telegram bot, email и другие. Позволяет гибко настроить инструмент для реализации практически любой логики обмена информацией между различными источниками

- Возможность управления запуском активаторов на распределённых кластерах GIMS Automation

Активаторы

[Добавить](#) [Удалить](#)

Имя	Тип	Кластер	Состояние
CASE_4	Periodic	Шина данных	⊙
CASE_5	Periodic	Шина данных	⊙
CASE_Periodic	Periodic	Шина данных	⊙
CASE_put_HTTP	HTTP	Шина данных	⊙
CSV_HDFS_periodic	Schedule	Шина данных	⊙
Datasources_Discovery_Test	Datasources Discovery	Шина данных	⊙
FaultDB_Enrichment	Enrichment	Шина данных	⊙
HTTP_aiohttp	HTTP aiohttp	Шина данных	⊙
HTTP_TEST_CASE_01	HTTP	Шина данных	⊙
HTTP_CASE_1	HTTP	Шина данных	⊙
HTTP_CASE_2	HTTP	Шина данных	⊙
HTTP_CASE_3	HTTP	Шина данных	⊙
Inventory_put_test	Schedule	Шина данных	⊙
mirapolis	Mirapolis HTTP	Шина данных	⊙
mirapolis_old	HTTP	Шина данных	⊙
Network Discovery Test	Network Discovery	Шина данных	⊙
NKO_sync	Periodic	Шина данных	⊙
RabbitMQ_HDFS	RabbitMQ	Шина данных	⊙
RabbitMQ_test	RabbitMQ	Шина данных	⊙

HTTP_CASE_1 x

Тип активатора: HTTP

Кластер: Шина данных

Сценарий: CASE_1

Таймаут выполнения сценария, сек: 61

Аутентификация

username: test

password:

Сетевые настройки

port: 8088

Журнал выполнения активатора



- Инструментарий GIMS Automation позволяет создавать собственные шаблоны триггеров для решения специфических задач
- Создание пользовательских шаблонов триггеров осуществляется в web редакторе и не требует существенных усилий

Типы активаторов

Добавить Удалить

Имя

- Cron_Sync_DB
- Datasources Discovery
- Enrichment
- HTTP
- HTTP aiohttp
- HTTP aiohttp gunicorn
- HTTP aiohttp limit
- HTTP_monitor
- mail_monitor
- Mirapolis HTTP
- Network Discovery
- Periodic
- RabbitMQ
- Schedule
- Script Runner
- SNMP trap listener
- SNMP Trap Listener 1.1
- Telegram Bot
- Telegram Bot Test
- test
- Zabbix
- Zabbix_Sync_Alarms
- Zabbix_Sync_History
- Zabbix_Sync_Inventory

HTTP x

Описание Свойства Программный код

Добавить Удалить

<input type="checkbox"/>	Имя (eng) *	Описание	Тип *	Значение по умолчанию	Раздел *	Обязательный	Скрытый
<input type="checkbox"/>	password	Пароль	Строковое	*****	Аутентификация	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/>	username	Имя пользователя	Строковое	Значение по умолчанию	Аутентификация	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	timeout	Таймаут подключения (сек)	Целое	5	Сетевые настройки	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	port	Порт	Целое	Значение по умолчанию	Сетевые настройки	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	request_type	Тип запроса	Справочник	GET	Параметры обрабатыва...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	content_type	Формат входных данных	Справочник	plain/text	Параметры обрабатыва...	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	content_type_out	Формат выходных данных	Справочник	plain/text	Параметры обрабатыва...	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- Замещение западных систем автоматизации ETL качественным российским продуктом, функционально превышающим большинство аналогов
- Возможность быстрого внедрения инструмента за счет высокой степени автоматизации инструмента
- Отсутствие необходимости держать в штате (или привлекать с рынка) высококвалифицированных и дорогостоящих специалистов для поддержки и сопровождения компонентов решения
- Техническая поддержка и обучение сотрудников для более быстрой адаптации продукта в инфраструктуре клиента
- Возможность многократного использования инструмента при реализации различных сценариев применения

GIMS Automation – это полноценная замена западных ETL систем и open source решений

Informatica PowerCenter

IBM WebSphere DataStage

Tibco

WSO2 ESB

Pentaho

Oracle Service Bus

Инвентаризация инфраструктуры

- создание выносных кластеров для сбора данных из изолированных сегментов сети
- интеграция с уже существующими системами учёта для объединения данных

Зонтичная система мониторинга

- создание выносных кластеров для сбора данных из изолированных сегментов сети
- автоматически сбор инвентарной информации об инфраструктуре заказчика с целью дальнейшего использования инвентарных данных для корреляции событий и построения топологии инфраструктуры
- сбор событий и метрик производительности от объектов мониторинга

Корпоративное хранилище данных

- обеспечение преобразования и объединения данных из нескольких источников для дальнейшего сохранения в централизованное хранилище
- создание выносных кластеров для сбора данных из изолированных сегментов сети
- автоматический сбор метаданных о структуре источников, хранилища, оборудования и программного обеспечения с целью дальнейшего представления всех слоёв хранилища данных в виде топологии

- Значительное снижение технологических, финансовых и страховых рисков клиентов за счет высокого качества ПО и лояльной ценовой политики
- **Отсутствие санкционного давления**
- Соответствие требованиям политики импортозамещения и программе цифровой трансформации бизнеса
- **Бесшовное и поэтапное внедрение** решений за счет возможности осуществлять техническую поддержку программных продуктов западных вендоров без потерь в производительности инфраструктуры
- Моделирование и реализация любых сценариев и налаживание любых бизнес и ИТ-процессов в соответствии с потребностями клиента
- Отсутствие необходимости держать в штате (или привлекать с рынка) высококвалифицированных и дорогостоящих специалистов для поддержки и сопровождения систем
- **Наличие единого инструмента** для инвентаризации и сбора данных, автоматизации ETL и создания консолидированной отчетности
- **Высочайшая нагрузочная способность** за счет автоматизированного механизма горизонтального масштабирования

Спасибо за внимание!

Телефон: +7 (495) 008-67-21

E-mail: partner_support@gelarm.ru