

1. Общие сведения о системе

1.1. Назначение системы

Система «КПП» (Контроль посторонних предметов) предназначена для автоматизированного выявления посторонних предметов на подвижном составе с использованием машинного зрения и компьютерного обучения. Система обрабатывает изображения коммерческого осмотра и камер ЛКПС, далее формирует отчёты о выявленных опасных зонах (лежни и крепление пояса цистерны и рамы).

1.2. Место системы в архитектуре ППСС

Система «КПП» **является модулем системы ППСС и встраивается в существующую инфраструктуру ППСС.** Полноценное функционирование возможно **только в составе системы ППСС.** Либо требуется развёртывание стендового окружения, имитирующего систему ППСС.

Модуль «КПП»:

- развёртывается в существующем кластере **Consul–Nomad** системы ППСС;
- подключается к **RabbitMQ** ППСС для получения заданий и публикации результатов;
- использует **MongoDB** ППСС для хранения служебных данных;
- получает изображения для обработки через **DbService** ППСС (URL изображений поступают через RabbitMQ);
- взаимодействует с центральным backend-сервисом ППСС **InfoProvider**;
- регистрируется в Consul как сервис `KppHttpProvider` .

1.3. Регистрация в Consul

При запуске модуль «КПП» автоматически регистрируется в Consul со следующими параметрами:

Параметр	Значение
Name	<code>KppHttpProvider</code>
ServiceAddress	<code>http://localhost:{Port}</code>

где `Port` — порт сервиса, задаваемый в конфигурации (см. раздел 4).

2. Минимальные системные требования

Параметр	Значение
Режим работы	Непрерывный
Класс условий размещения по воздействиям механических нагрузок по ГОСТ 34012-2016	МС1
Группа климатического исполнения по ГОСТ 15150-69	УХЛ4.2
Класс условий размещения по воздействиям климатических факторов по ГОСТ 34012-2016	К1.1
Номинальное напряжение питания, В, при частоте 50 Гц	220
Потребляемая мощность 50 кВт, не более	2
Количество процессоров, шт.	2
Блок обработки данных	
Операционная система	Astra Linux*
Частота процессора, ГГц, не менее 2.9	2,9
Количество ядер, шт., не менее	16
Количество потоков, шт., не менее	32
Объём оперативной памяти, Гбайт, не менее	128

Объём видеопамяти, Тбайт, не менее	24
Частота работы видеочипа, ГГц	2,1
Объём SSD, Тбайт, не менее	2
Примечание - * допускается применение другой операционной системы, включенной в реестр российского программного обеспечения и стандартных средств операционных систем, имеющим допуск к применению в ОАО "РЖД"	

2.1. Требуемое предустановленное ПО

На сервере должны быть установлены и настроены:

- **Docker** (актуальная стабильная версия);
- **Nomad** (в составе кластера ППСС);
- **Consul** (в составе кластера ППСС);
- **NVIDIA drivers** (версия, совместимая с видеокартами);
- **nvidia-docker2** (для работы Docker-контейнеров с GPU).

3. Состав дистрибутива

Дистрибутив поставляется в виде архива `kpp-distribution.tgz`, скрипта, выполняющего развёртывание и документации.

Скрипт распакует архив, загрузит Docker-образы в локальный реестр, запустит три Nomad-job, соответствующие трём подсистемам КПП.

4. Запуск автоматической установки

Запустите скрипт установки:

```
chmod +x install.sh
./install.sh
```

Скрипт выполняет следующие действия:

1. распаковывает архив (если это не было сделано вручную);
2. загружает Docker-образы;
3. регистрирует шаблоны конфигураций в Consul KV;
4. запускает Nomad-джобы `kpp_bogie_beam`, `kpp_bogie_frame`, `kpp_cistern_ip`.

4.5. Первичная настройка конфигураций

При первом запуске в Consul автоматически создаются конфигурации по умолчанию. **Перед началом работы необходимо скорректировать параметр `RabbitMqConnectionString`** — указать адрес RabbitMQ системы ППСС.

Конфигурации хранятся в Consul KV и доступны для редактирования через веб-интерфейс Consul или CLI. Пример структуры конфигурации для каждой джобы приведён ниже.

4.5.1. Конфигурация для балки тележки (IP.BogieBeam.json)

```
{
  "ServiceBus": {
    "RabbitMqConnectionString": "amqp://<пользователь>:<пароль>@<адрес>:5672/",
    "RebusStoragePath": "RebusStorage",
    "ManagementPort": 15672,
    "LinkedVirtualHosts": []
  },
  "BatchSize": 6,
  "NumAnalyzers": 4,
  "WrapperPrometheusPort": 7057,
  "KnnNeighbors": 11,
  "DetectorThresholds": {
    "LowThreshold": 0.35,
    "ModerateThreshold": 0.4,
    "HighThreshold": 0.5
  }
}
```

```
},
"BeamInnerKnnThresholds": {
  "LowThreshold": 0.85,
  "ModerateThreshold": 0.9,
  "HighThreshold": 0.99
}
}
```

Обязательно измените:

- RabbitMqConnectionString — на реальный адрес RabbitMQ ППСС;
- ManagementPort — при необходимости;
- порты Prometheus (WrapperPrometheusPort) — если они конфликтуют с другими сервисами.

4.5.2. Конфигурация для рамы тележки (IP.BogieFrame.json)

```
{
  "ServiceBus": {
    "RabbitMqConnectionString": "amqp://<пользователь>:<пароль>@<адрес>:5672/",
    "RebusStoragePath": "RebusStorage",
    "ManagementPort": 15672,
    "LinkedVirtualHosts": []
  },
  "BatchSize": 12,
  "NumAnalyzers": 4,
  "WrapperPrometheusPort": 7058,
  "KnnNeighbors": 11,
  "FrameKnnThresholds": {
    "LowThreshold": 0.85,
    "ModerateThreshold": 0.9,
    "HighThreshold": 0.99
  }
}
```

Обязательно измените: RabbitMqConnectionString .

4.5.3. Конфигурация для цистерн (IPCistern.json)

```
{
  "ServiceBus": {
    "RabbitMqConnectionString": "amqp://<пользователь>:<пароль>@<адрес>:5672/",
    "RebusStoragePath": "RebusStorage",
    "ManagementPort": 15672,
    "LinkedVirtualHosts": []
  },
  "BatchSize": 18,
  "NumAnalyzers": 4,
  "WrapperPrometheusPort": 7059,
  "UseDetector": true,
  "UseKnn": false,
  "KnnNeighbors": 11,
  "DetectorThresholds": {
    "LowThreshold": 0.35,
    "ModerateThreshold": 0.5,
    "HighThreshold": 0.75
  },
  "FastenerMedianCosineThresholds": {
    "LowThreshold": 0.01,
    "ModerateThreshold": 0.1,
    "HighThreshold": 0.49
  },
  "BeamMedianCosineThresholds": {
    "LowThreshold": 0.04,
    "ModerateThreshold": 0.1,
    "HighThreshold": 0.49
  },
  "BeamKnnThresholds": {
```

```
"LowThreshold": 0.1,
"ModerateThreshold": 0.3,
"HighThreshold": 0.6
},
"FastenerKnnThresholds": {
  "LowThreshold": 0.1,
  "ModerateThreshold": 0.3,
  "HighThreshold": 0.6
}
}
```

Обязательно измените: `RabbitMQConnectionString` .

4.6. Перезапуск джоб после изменения конфигураций

После внесения изменений в конфигурации в Consul KV перезапустите Nomad-джобы:

```
nomad job restart kpp_bogie_beam
nomad job restart kpp_bogie_frame
nomad job restart kpp_cistern_ip
```

Либо выполните повторный запуск через CLI:

```
nomad job run -force /opt/kpp/kpp_bogie_beam.hcl
nomad job run -force /opt/kpp/kpp_bogie_frame.hcl
nomad job run -force /opt/kpp/kpp_cistern_ip.hcl
```

4.7. Проверка статуса джоб

Убедитесь, что все три джобы запущены и работают:

```
nomad job status kpp_bogie_beam
nomad job status kpp_bogie_frame
nomad job status kpp_cistern_ip
```

В выводе должны быть указаны статус `running` и выделенные ресурсы.

5. Интеграция с системой ППСС

5.1. Требуемые внешние подключения

Для полноценной работы модуля «КПП» необходимы:

Компонент ППСС	Назначение
RabbitMQ ППСС	Получение заданий и публикация результатов обработки
MongoDB ППСС	Хранение служебных данных и результатов
DbService ППСС	Получение URL изображений для обработки
InfoProvider	Центральный backend-сервис ППСС, взаимодействие через API
KppFlair	Сервис формирования тревог КПП на стороне ППСС
Consul кластер	Service discovery, хранение конфигураций
Nomad кластер	Оркестрация контейнеров

5.2. Входящие сообщения (задания)

Модуль «КПП» получает из RabbitMQ следующие типы сообщений:

Сообщение	Описание
MsgLazerStockImageSetReady	Изображения коммерческого осмотра готовы

Сообщение	Описание
MsgKppCisternTask	Задание на обработку крепежей и лежней того или как тут Результаты обработки цистерны (крепёж, лежни)

Пример структуры входящего сообщения (задание на обработку балки тележки):

```
public class MsgKppBogieBeamTask
{
    public long TrainId { get; set; }
    public int WagonIndex { get; set; }
    public WagonImageInformation[] WagonImageInformationArray { get; set; } = [];
}

public class WagonImageInformation
{
    public string ImageUrl { get; set; } = string.Empty;
    public KppWagonSide WagonSide { get; set; }
    public KppSequentialNumber KppSequentialNumber { get; set; }
}
```

5.3. Исходящие сообщения (результаты обработки)

По результатам обработки модуль «КПП» публикует в RabbitMQ следующие сообщения:

Сообщение	Описание
MsgKppCisternResult	Результаты обработки цистерны (лежни и крепление пояса цистерны и рамы)

Пример структуры результата:

```
public class MsgKppBogieBeamResult
{
    public long TrainId { get; set; }
    public int WagonIndex { get; set; }
    public KppBogieBeamZoneResult[] ZoneResults { get; set; } = [];
}

public class KppBogieBeamZoneResult
{
    public KppWagonSide WagonSide { get; set; }
    public KppSequentialNumber KppSequentialNumber { get; set; }
    public KppDangerZona KppDangerZona { get; set; }
    public string? ImgUrl { get; set; } = string.Empty;
    public KppDangerLevel DangerLevel { get; set; }
    public float X { get; set; }
    public float Y { get; set; }
    public float Width { get; set; }
    public float Height { get; set; }
}
```

6. Проверка работоспособности

После запуска и настройки выполните проверку:

1. Проверка регистрации в Consul:

```
consul catalog services | grep KppHttpProvider
```

Сервис `KppHttpProvider` должен присутствовать в списке.

2. Проверка Nomad-джоб:

```
nomad job status
```

Все три джобы (`kpp_bogie_beam` , `kpp_bogie_frame` , `kpp_cistern_ip`) должны иметь статус `running` .

3. **Проверка подключения к RabbitMQ:** Убедитесь, что в логах контейнеров нет ошибок подключения к RabbitMQ:

```
docker logs <container_id>
```

4. **Проверка публикации сообщений:** Отправьте тестовое задание через RabbitMQ (или дождитесь поступления реального задания) и убедитесь, что в очереди результатов появилось сообщение `MsgKppBogieBeamResult` , `MsgKppBogieFrameResult` или `MsgKppCisternResult` .

Comments



tutikoff @tutikoff · 1 week ago

Developer

1.1 Так не правильнее будет?: машинного зрения и компьютерного обучения.

ТК и ФК на текущий момент не реализовано. Сейчас берем изображения с камер ЛКПС.

Отчёты о выявленных опасных зонах (лежни и крепление пояса цистерны и рамы).

2. Минимальные системные требования из технического задания:

3. Состав дистрибутива - почему 3 КПП?

5.2. Входящие сообщения (задания): Изображения колодки загружены в базу - не реализовано и по ТЗ не заявлено. Изображения клина загружены в базу - не реализовано и по ТЗ не заявлено.

Пример структуры входящего сообщения (задание на обработку балки тележки): у нас лежни и крепление пояса цистерны и рамы.

5.3 лежни и крепление пояса цистерны и рамы.



gorbov @gorbov · 1 week ago

Maintainer

Состав дистрибутива - почему 3 КПП?

Дистрибутив состоит из одного архива и скрипта который всё это дело развернёт. Три джоба в номад - потому что так удобнее описывать и управлять. Это нормально. В ППССах их по 50 штук



tutikoff @tutikoff · 1 week ago

Developer

Принимается!

Reply...