DevOps meets BigData

Kafka im Versicherungsumfeld
Welcome to our talk ;)

Who we are:

Julian Schreiner
BigData DevOps @ HUK
Cloudera Hadoop, Kafka, Infrastructure, Linux, Ansible/Puppet, K8s

Bernhard Hopfenmüller
SeniorITConsultant@ ATIX
DevOps, Ansible, Kafka, Container, Pipelines
BigData Environment for Telematics / DataLake / Event Platform

Transition from on-premise ETL/Batch processing to near-realtime event-platform and hybrid-cloud.

Cloud

- HUK AWS
  - Trip Splitting
  - Scoring
  - Crash Detection

DMZ

- Kafkak

On-premise

- HUK Streaming- und Eventplattform
  - Realtime-Verarbeitung von Events aus Online Angebotsdaten für Analytics und Batch

HUK BigData Software Stack

- Hadoop
- HBase
- Spark

Telematik App

- Meine Gesundheit

KFZ Tripdaten

Näher am Kunden durch BigData!
Kafka in a nutshell

Everything clear? ;)

Kafka in a nutshell

Kafka is...
- a distributed, horizontally-scalable, highly available, fault-tolerant commit log
- a publish-subscribe messaging system
- a Streaming Data platform

(Confluent) Kafka is complex

- Installing Zookeeper + Broker + Healthchecks: > 100 steps
- More components on top
- Kafka Broker: > 200 config settings
- Standardized & automated installation
- Attention (Watch out anyway!)
Ansible

- Config management and automation tool
- Clientless
- Idempotence: describe the final state
- Parallel execution on all hosts by default
- Execute specific tasks on a group of hosts

1. YAML → Python Code
2./4. SSH/Ansible
3. Create desired state with Python Code
Kafka meets Ansible

Confluent Platform-Ansible:
https://github.com/confluentinc/cp-ansible

Open Source Playbooks to install Confluent Kafka

Assistance for Automated Security Setup

By now: Rolling update/upgrade of Clusters

Ideally: Only manage inventory, use roles from upstream
Excerpt from Inventory

kafka_broker:
  vars:
    kafka_broker_kerberos_keytab_path: "keytabs/kafka_{{{inventory_hostname_short}}}.keytab"
    kafka_broker_kerberos_principal: "kafka/{{inventory_hostname}}@NETWORK.HUK-COBURG.DE"
    kafka_broker_keytab_path: "*/etc/security/keytabs/kafka_{{{inventory_hostname_short}}}.keytab"
    kafka_broker_service_environment_overrides:
      KAFKA_HEAP_OPTS: "-Xms6G -Xmx6G -XX:MetaspaceSize=96m -XX:+UseG1GC -XX:MaxGCPauseMillis=20"

kafka_broker:
  datadir:
    - /dfs1/kafka/data
    - /dfs2/kafka/data

hosts:
  broker1.network.huk-coburg.de:
    broker_id: 1
    kafka_broker_custom_properties:
      broker.rack: DC1

  broker2.network.huk-coburg.de:
    broker_id: 2
    kafka_broker_custom_properties:
      broker.rack: DC2

  broker3.huk-coburg.de:
    broker_id: 3
    kafka_broker_custom_properties:
      broker.rack: DC3
Architecture and Setup
Making Kafka Cluster HA - Duplex vs Stretched Cluster

Double lambda architecture

Confluent stretched cluster

Ping between DCs < 0.5 ms
Architecture

Components:
- 5 Zookeeper Server
- 9 Kafka Broker
- Confluent Schema Registry
- Confluent Kafka Connect
- Confluent Control Center
Building a PoC Cluster

Steps:
- Setup without Security ✓
- Switch on TLS ✓
- Enable Kerberos ✗
- Enable Role Based Access Control (via LDAP) ✗
Kerberos Error Messages

Failure unspecified at GSS-API level
(Mechanism level: Checksum failed)

No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt)

→ Among other reasons: keytab not there or somehow broken
More Kerberos Error Messages

javax.security.auth.login.LoginException: No password provided
javax.security.auth.login.LoginException: Unable to obtain password from user

- Wrong keytab file
- Wrong entry in keytab
- Principal name wrong
- Hostname missmatch
Kerberos
"Error Messages To Fear"

Automate Setup as far as possible!

https://github.com/steveloughran/kerberos_and_hadoop/blob/master/sections/errors.md
Creating Kerberos Principals via Ansible

- name: Write LDIF File for service {{item.0.name}} and ID {{item[1]}}
  copy:
    dest: "add_user_new_{{item.0.name}}_{{item[1]}}.ldif"
    content: |
      dn: CN={{item.0.name}}/{{server_prefix}}{{item[1]}}.{{domain_name}},{{target_ou}}
      changetype: add
      objectClass: top
      objectClass: person
      objectClass: organizationalPerson
      objectClass: user
      distinguishedName: CN={{item.0.name}}/{{server_prefix}}{{item[1]}}.{{domain_name}},{{target_ou}}
      userAccountControl: 66048
      accountExpires: 0
      sAMAccountName: {{samaccount_prefix}}{{item.0.name [:3] }}{{server_prefix}}{{item[1]}}
      sn: {{cluster_name}}
      description: {{description}}
      userPrincipalName: {{item.0.name}}/{{server_prefix}}{{item[1]}}.{{domain_name}}{{realm}}
      servicePrincipalName: {{item.0.name}}/{{server_prefix}}{{item[1]}}.{{domain_name}}
      unicodePwd::={{ passhash }}

- name: Creates arcfour-hmac cipher keytab file
  shell:
    - "printf "%b\"' "addent -password -p {{item.0.name}}/{{server_prefix}}{{item[1]}}.{{domain_name}}{{realm}}
    -k 1 -e arcfour-hmac\n{{ password }}\nwrite_kt {{item.0.name}}_{{server_prefix}}{{item[1]}}.keytab\" | kttutil"
Performance Tests with Ansible
Producer Performance Results

runner_cluster1-producer
runner_cluster2-producer

<table>
<thead>
<tr>
<th>Records sent(k)</th>
<th>Records/s</th>
<th>MB/s</th>
<th>Avg. Latency(ms)</th>
<th>Max. Latency(ms)</th>
<th>50th %ile(ms)</th>
<th>95th %ile(ms)</th>
<th>99th %ile(ms)</th>
<th>99.9th %ile(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>3053</td>
<td>1429</td>
<td>20</td>
<td>1390</td>
<td>2850</td>
<td>3503</td>
<td>4011</td>
<td>4983</td>
</tr>
</tbody>
</table>
Resilience Tests

among other things:

- Kill servers in operation
- Kill network between DCs
- Pull disks in operation
- Split-brain Zookeeper Cluster
Well prepared!

Or not?
Whoopsie – Cluster Up, but not working?!


→ Broker in "Zombie"- State
Stress Testing Kafka Cluster

"Only a cluster that breaks during the stress test is a healthy cluster."

<table>
<thead>
<tr>
<th>Artificially induced network latency (tc-netem)</th>
<th>Artificially induced CPU/Memory pressure (stress-ng)</th>
<th>Artificially slowed disk (dm-delay)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker ping: 0.15 ms -&gt; 10ms:</td>
<td>No major implications (after increasing HEAP)</td>
<td>No major implications (up to 200ms)</td>
</tr>
<tr>
<td>massive amount of (partially) unavailable partitions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Whoopsie Again

Test Cluster -> higher version

Topic locations are wrong

Throw away in between data

Cluster not working

All topics empty

Parameter for topic location wrong

Thank god we have a test cluster :)

All permissions lost

Topics were new created in default location

Upgrading Cluster

Test Cluster not working

All topics empty

Parameter for topic location wrong

Thank god we have a test cluster :)

All permissions lost

Topics were new created in default location
Be prepared

- Be careful with Automation – scaling errors!
- Read the manual
- Test upgrades!
- Be prepared for failure
Lessons Learned

Automate and Test all you can

Latency is crucial!

Break it to understand it

It will fail eventually, are you prepared for that?
How we make our lives easier - Kafka Operations

- Server infrastructure
  - puppet
  - HUK Modules

- Server updates
  - Ansible
  - HUK Playbook

- Kafka cluster
  - Ansible
  - cp-ansible

- Kafka resources
  - Julie-ops
  - Jenkins
  - HUK Bash scripts ;)

DevOps meets BigData
Daily Doings

- Topic creation
- Granting permissions
- Deploying Connectors

Kafka Topology Builder

Jenkins
Kafka Resources - Topics

- Tool: julie-ops, former kafka topology builder
  - Entwickler: Pere Urbon-Bayes
  - Github: https://github.com/kafka-ops/julie

- Topic description via topology file (yaml manifest)
  - Topics
  - Permissions

- Enabling multitenancy

```
---
context: "ib13"
source: "dev"
projects:
- name: "training"
  consumers:
    - principal: "User:Julian"
  producers:
    - principal: "User:Julian"
  topics:
    - name: "julian"
      config:
        replication.factor: "3"
        num.partitions: "6"
        retention.ms: "3600000"

Topic name:
ib13.dev.training.julian

Granting RBAC permissions

Defining topic configurations
```

15.10.21  DevOps meets BigData
Julie-ops

<table>
<thead>
<tr>
<th>Love</th>
<th>Hate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idempotence</td>
<td>Missing features</td>
</tr>
<tr>
<td>Enabling a topic naming convention</td>
<td>No official support</td>
</tr>
<tr>
<td>Uses kafka tools</td>
<td>Waiting for fixes</td>
</tr>
<tr>
<td>GitOps approach - cluster resource and permission management in git</td>
<td>Evaluation of provisioning based on ansible and Kafka REST API</td>
</tr>
<tr>
<td>Automation via jenkins</td>
<td></td>
</tr>
</tbody>
</table>
Kafka Resources – Connectors

- Usage of Kafka Connect REST API for Connector deployment
- Version control of existing connectors in git
- Automated deployment via Jenkins
HUK Telematics behind the scenes

- Kafka as central data pipeline
- Data processing via batch and streaming jobs
- Focus on high availability
Failover concept

- Usage maintenance jobs, e.g. major upgrades
- Transparent access for clients
  - Loadbalancer as bootstrap server enables transparent failover
- Replication of messages to passive kafka cluster
  - Usage of kafka message header to prevent loop replication
Failover mechanism

- Switch to other cluster via ansible playbook
  1. Switch loadbalancer to passive cluster
  2. Stop and delete replication from dmz to active cluster
  3. Remove permissions on topics on active cluster (clients fail)
  4. Create permissions on topics on passive cluster
  5. Start replication from dmz to passive cluster
Monitoring

- Monitoring and alerting via Prometheus
  - Server monitoring via node-exporter
  - Kafka monitoring via jmx metrics
  - Failover monitoring via health-check metrics

- Visualization in Grafana Dashboards

- Log management via Splunk
Dashboards

- Cluster overview
- Monitoring of kafka components and cluster traffic
Alerting

- Prometheus Alerts for critical situations, e.g. server down, kafka problems, …
- Forwarding to central event management

<table>
<thead>
<tr>
<th>Server:</th>
<th>Broker 1</th>
<th>(ServerDB-Status: Aktiv-Aktiv, Dashboard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem:</td>
<td>Kafka-Cluster LANKafka1: Broker Topics mit UnderReplicated Partitionen (AppKafkaBrokerUnderreplicatedpartitions)</td>
<td></td>
</tr>
<tr>
<td>Problem-Zustand:</td>
<td>WARNUNG</td>
<td></td>
</tr>
<tr>
<td>Meldung seit:</td>
<td>08.07.2021 11:43:13</td>
<td></td>
</tr>
<tr>
<td>Details:</td>
<td>Im Cluster LANKafka1 meldet Broker 1 7 underreplicated Partitionen</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Server:</th>
<th>Broker 1</th>
<th>(ServerDB-Status: Aktiv-Aktiv, Dashboard)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem:</td>
<td>Kafka-Cluster LANKafka1: Broker Topics mit UnderReplicated Partitionen (AppKafkaBrokerUnderreplicatedpartitions)</td>
<td></td>
</tr>
<tr>
<td>Problem-Zustand:</td>
<td>⬇️ BEHOBEN</td>
<td></td>
</tr>
<tr>
<td>Meldung seit:</td>
<td>08.07.2021 11:43:13</td>
<td></td>
</tr>
<tr>
<td>Behoben seit:</td>
<td>08.07.2021 11:51:13</td>
<td></td>
</tr>
</tbody>
</table>
Automated server updates

- Rolling updates via Ansible Playbook
  - Run health checks
  - Stopping services on host
  - Update server
  - Starting services
  - Run health check

tasks:
- name: Kafka Health Check
  import_role:
    name: confluent.kafka_broker
  tasks_from: health_check.yml
  when: not ansible_check_mode
  tags: kafka_broker_restart

- name: Stop confluent-server.service before huk-server-update systemd:
  name: "{{ kafka_broker_service_name }}"
  state: stopped
  tags: kafka_broker_restart

- name: Execute huk-server-update Script
  command: /sbin/huk-server-update --batch --force --no-reboot
  become: true
  register: huk_server_update_result
  failed_when: huk_server_update_result.rc != 0
  - debug:
    var: huk_server_update_result.rc

- name: Reboot server
  reboot:
    msg: "Reboot initiated by Ansible"
    connect_timeout: 5
    reboot_timeout: 600
    pre_reboot_delay: 0
    post_reboot_delay: 30
    test_command: whoami

- name: Ensure Kafka Broker Service is running systemd:
  name: "{{ kafka_broker_service_name }}"
  state: started
  tags: kafka_broker_restart

- name: Kafka Health Check
  import_role:
    name: confluent.kafka_broker
  tasks_from: health_check.yml
  when: not ansible_check_mode
  tags: kafka_broker_restart

Automated server updates
Rolling updates via Ansible Playbook
- Run health checks
- Stopping services on host
- Update server
- Starting services
- Run health check
What else

- Self-Service – Kafka Clusters
- GitOps – Pipelines – Infrastructure/Config as Code
Multi Environment – Configure new cluster

key/value + Pipeline

→ Use Kafka itself
Kafka as central event hub

Cloud
- HUK AWS
  - Trip Splitting
  - Scoring
  - Crash Detection

DMZ
-.kafka-

On-premise
- HUK Streaming- und Eventplattform
  - Realtime-Verarbeitung von Events aus Online Angebotsdaten für Analytics und Batch

Näher am Kunden durch BigData!

KFZ Tripdaten

Telematik App
- Meine Gesundheit

HUK BigData Software Stack
- Apache Hadoop
- Apache HBase
- Apache Spark

Backup/Archive
- Database
- Analytics
- Datalake
- ETL
Thanks for your attention!

Bernhard Hopfenmüller
LinkedIn/Xing
hopfenmueller@atix.de
Twitter/Github/Matrix: Fobhep

Julian Schreiner
julian.schreiner@huk-coburg.de