Конфигурация NOC

Получение настроек NOC

У NOC довольно большое количество настроек. Получить текущее состояние можно командой:

```
./noc config dump
```

```
activator:
  buffer_size: 1048576
  connect_retries: 3
  connect_timeout: 3
  http_connect_timeout: 20
  http_request_timeout: 30
  http_validate_cert: false
  script_threads: 10
  tos: 0
audit:
  command_ttl: 1m
  config_changed_ttl: 1y
  config_ttl: 1y
  db_ttl: 5y
  login_ttl: 1m
  reboot_ttl: 0
backup:
  keep_day_of_month: 1
  keep_day_of_week: 6
  keep_days: 2w
  keep_months: 12
  keep_weeks: 12w
bi:
  alarms_archive_batch_limit: 10000
  alarms_archive_policy: weekly
  chunk_size: 3000
  clean_delay_alarms: 1d
  clean_delay_reboots: 1d
  enable_alarms: false
  enable_alarms_archive: false
  enable_managedobjects: false
  enable_reboots: false
  extract_delay_alarms: 1h
  extract_delay_reboots: 1h
  extract_window: 1d
  language: en
  query_threads: 10
  reboot_interval: 1M
brand: NOC
cache:
  cache_class: noc.core.cache.redis.RedisCache
  default_ttl: 1d
  pool_size: 8
  vcinterfacescount: 1h
  vcprefixes: 1h
```

```
card:
  alarmheat_tooltip_limit: 5
  language: en
chwriter:
  batch_delay_ms: 10000
  batch_size: 50000
  channel_expire_interval: 5M
  max_in_flight: 10
  records_buffer: 1000000
  suspend_timeout_ms: 3000
  topic: chwriter
classifier:
  default_interface_profile: default
  default_rule: Unknown | Default
  lookup_handler: noc.services.classifier.rulelookup.RuleLookup
clickhouse:
  cluster_topology: 1
  connect_timeout: 10s
  db: noc
  default_merge_tree_granularity: 8192
  enable_low_cardinality: false
  encoding: null
  request_timeout: 1h
  ro_addresses: clickhouse:8123
  ro_user: readonly
  rw_addresses: clickhouse:8123
  rw_user: default
collections:
  allow_sharing: true
consul:
  base: noc
  check_interval: 10s
  check_timeout: 1s
  connect_timeout: 5s
  host: consul
  keepalive_attempts: 5
  lock_delay: 20s
  near_retry_timeout: 1
  port: 8500
  release: 1M
  request_timeout: 1h
  retry_timeout: 1s
  session_ttl: 10s
correlator:
  auto_escalation: true
  discovery_delay: 10M
  max_threads: 20
  oo_close_delay: 20s
  topology_rca_window: 0
customization:
  branding_background_color: null
  branding_color: null
  favicon_url: /ui/web/img/logo_24x24_deep_azure.png
  logo_height: 24
  logo_url: /ui/web/img/logo_white.svg
  logo_width: 24
  preview_theme: midnight
datasource:
  chunk_size: 1000
  default_ttl: 1h
  max_threads: 10
datastream:
  enable_address: false
  enable_address_wait: true
  enable_administrativedomain: false
  enable_administrativedomain_wait: true
  enable_alarm: false
  enable_alarm_wait: true
  enable_cfgping: true
  enable_cfgping_wait: true
  enable_cfgsyslog: true
```

```
  enable_cfgsyslog_wait: true
  enable_cfgtrap: true
  enable_cfgtrap_wait: true
  enable_dnszone: false
  enable_dnszone_wait: true
  enable_managedobject: false
  enable_managedobject_wait: true
  enable_prefix: false
  enable_prefix_wait: true
  enable_resourcegroup: false
  enable_resourcegroup_wait: true
  enable_vrf: false
  enable_vrf_wait: true
dcs:
  resolution_timeout: 5M
  resolver_expiration_timeout: 10M
discovery:
  max_threads: 20
  sample: 0
dns:
  warn_before_expired: 1m
escalator:
  ets: 1M
  max_threads: 5
  retry_timeout: 1M
  sample: 0
  tt_escalation_limit: 10
  wait_tt_check_interval: 1M
features:
  consul_healthchecks: true
  cp: true
  cpclient: false
  forensic: false
  sentry: false
  service_registration: true
  telemetry: false
  traefik: false
  use_uvlib: false
fm:
  active_window: 1d
  alarm_close_retries: 5
  keep_events_with_alarm: -1
  keep_events_wo_alarm: 0
  outage_refresh: 1M
  total_outage_refresh: 1M
geocoding:
  google_key: null
  google_language: en
  negative_ttl: 1w
  order: yandex,google
  yandex_apikey: null
  yandex_key: null
gis:
  ellipsoid: PZ-90
  enable_google_roadmap: false
  enable_google_sat: false
  enable_osm: true
  tile_size: 256
  tilecache_padding: 0
global_n_instances: 1
grafanads:
  db_threads: 10
help:
  base_url: https://docs.getnoc.com
  branch: microservices
  language: en
http_client:
  buffer_size: 131072
  connect_timeout: 10s
  http_port: 80
  https_port: 443
```

```
  max_redirects: 5
  ns_cache_size: 1000
  request_timeout: 1h
  resolver_ttl: 3s
  user_agent: noc
  validate_certs: false
icqsender:
  retry_timeout: 2
  use_proxy: false
initial:
  admin_email: test@example.com
  admin_password: admin
  admin_user_name: admin
installation_id: null
installation_name: NOC-DC
instance: 0
language: en
language_code: en
layout:
  ring_chain_edge: 150
  ring_chain_spacing: 100
  ring_ring_edge: 150
  spring_bubble_force: 2.0
  spring_edge_force: 1.2
  spring_edge_spacing: 190
  spring_iterations: 50
  spring_propulsion_force: 1.5
  tree_horizontal_step: 100
  tree_max_levels: 4
  tree_vertical_step: 100
listen: auto:1200
log_format: \%(asctime)s [%(name)s] %(message)s
logging:
  log_api_calls: false
  log_sql_statements: false
login:
  idle_timeout: 1w
  language: en
  methods: local
  mutual_exclusive_group: null
  pam_service: noc
  radius_secret: noc
  register_last_login: true
  restrict_to_group: null
  session_ttl: 1w
  single_session_group: null
  user_cookie_ttl: 1
loglevel: info
mailsender:
  from_address: noc@example.com
  helo_hostname: noc
  smtp_port: 25
  use_tls: false
memcached:
  addresses: memcached
  default_ttl: 1d
  pool_size: 8
metrics:
  default_hist:
  - 0.001
  - 0.005
  - 0.01
  - 0.05
  - 0.5
  - 1.0
  - 5.0
  - 10.0
  default_quantiles:
  - 0.5
  - 0.9
  - 0.95
```

```
  enable_mongo_hist: false
  enable_mongo_quantiles: false
  enable_postgres_hist: false
  enable_postgres_quantiles: false
  mongo_hist:
  - 0.001
  - 0.005
  - 0.01
  - 0.05
  - 0.5
  - 1.0
  - 5.0
  - 10.0
  postgres_hist:
  - 0.001
  - 0.005
  - 0.01
  - 0.05
  - 0.5
  - 1.0
  - 5.0
  - 10.0
mongo:
  addresses: mongo:27017
  db: noc
  max_idle_time: 1M
  password: noc
  retries: 20
  retry_writes: false
  timeout: 3s
  user: noc
mrt:
  enable_command_logging: false
  max_concurrency: 50
nbi:
  max_threads: 10
  objectmetrics_max_interval: 3h
nsqd:
  addresses: nsqd:4150
  ch_chunk_size: 4000
  compression: null
  compression_level: 6
  connect_timeout: 3s
  http_addresses: nsqd:4151
  max_in_flight: 1
  mpub_messages: 10000
  mpub_size: 1048576
  pub_retries: 5
  pub_retry_delay: 1.0
  reconnect_interval: 15
  request_timeout: 30s
  topic_mpub_rate: 10
nsqlookupd:
  addresses: nsqlookupd:4160
  http_addresses: nsqlookupd:4161
path:
  babel: ./bin/pybabel
  babel_cfg: etc/babel.cfg
  backup_dir: /var/backup
  bi_data_prefix: /var/lib/noc/bi
  card_template_path: services/card/templates/card.html.j2
  collection_fm_mibs: collections/fm.mibs/
  cp_new: /var/lib/noc/cp/crashinfo/new
  custom_path: /opt/noc_custom
  cythonize: ./bin/cythonize
  etl_import: /var/lib/noc/import
  legacy_config: etc/noc.yml
  mib_path: /var/mib
  npkg_root: /var/lib/noc/var/pkg
  pm_templates: templates/ddash/
  pojson: ./bin/pojson
```

```
  repo: /var/repo
  ssh_key_prefix: etc/noc_ssh
  supervisor_cfg: etc/noc_services.conf
  vcs_path: /usr/local/bin/hg
peer:
  enable_arin: true
  enable_radb: true
  enable_ripe: true
  max_prefix_length: 24
  prefix_list_optimization: true
  prefix_list_optimization_threshold: 1000
  rpsl_inverse_pref_style: false
pg:
  addresses: postgres:5432
  connect_timeout: 5
  db: noc
  password: hqnl3OauEKT7wzn
  user: noc
ping:
  ds_limit: 1000
  receive_buffer: 4194304
  send_buffer: 4194304
  tos: 0
pmwriter:
  batch_size: 2500
  max_delay: 1.0
  metrics_buffer: 50000
  read_from: pmwriter
  write_to: influxdb
  write_to_port: 8086
pool: default
redis:
  addresses: redis:6379
  db: 0
  default_ttl: 1d
rpc:
  async_connect_timeout: 20s
  async_request_timeout: 1h
  retry_timeout: 0.1,0.5,1,3,10,30
  sync_connect_timeout: 20s
  sync_request_timeout: 1h
  sync_retries: 5
  sync_retry_delta: 2.0
  sync_retry_timeout: 1.0
sae:
  activator_resolution_retries: 5
  activator_resolution_timeout: 2s
  db_threads: 20
scheduler:
  autointervaljob_initial_submit_interval: 1d
  autointervaljob_interval: 1d
  cache_default_ttl: 1d
  max_chunk_factor: 1
  max_threads: 20
  submit_threshold_factor: 10
  updates_per_check: 4
script:
  caller_timeout: 1M
  calling_service: script
  session_idle_timeout: 1M
  timeout: 2M
secret_key: 12345
selfmon:
  enable_fm: true
  enable_inventory: true
  enable_managedobject: true
  enable_task: true
  fm_ttl: 30
  inventory_ttl: 30
  managedobject_ttl: 30
  task_ttl: 30
```

```
sentry:
  url: null
syslogcollector:
  ds_limit: 1000
  enable_freebind: false
  enable_reuseport: true
  listen: 0.0.0.0:514
tests:
  fixtures_paths:
  - tests/data
tgsender:
  retry_timeout: 2
  use_proxy: false
thread_stack_size: 0
threadpool:
  idle_timeout: 30s
  shutdown_timeout: 1M
timezone: Europe/Moscow
traceback:
  reverse: true
trapcollector:
  ds_limit: 1000
  enable_freebind: false
  enable_reuseport: true
  listen: 0.0.0.0:162
version_format: \%(version)s+%(branch)s.%(number)s.%(changeset)s
web:
  api_arch_alarm_limit: 345600
  api_row_limit: 0
  enable_remote_system_last_extract_info: false
  install_collection: false
  language: en
  macdb_window: 345600
  max_threads: 10
  max_upload_size: 16777216
  theme: gray
```

Можно получить конфигурацию только по одному компоненту

```
# ./noc config dump web
web:
  api_arch_alarm_limit: 345600
  api_row_limit: 0
  enable_remote_system_last_extract_info: false
  install_collection: false
  language: en
  macdb_window: 345600
  max_threads: 10
  max_upload_size: 16777216
  theme: gray
```

Порядок применения настроек

Настройки могут браться из разных источников и переопределятся на многих уровнях. Начальное конфигурирование делается в Tower и определяет порядок чтения настроек. По умолчанию настройки читаются в таком порядке `legacy:///,yaml:///opt/noc/etc/settings.yml,env:///NOC` Строка настроек читается с лева на право, настройки определенный в следующем источнике переопределяют предыдущий источник. Источники могут повторятся с указанием разных префиксов или суффиксов. В целом разбор строки производится по стандарту URI.

Немного подробнее про каждый вариант.

Источники настроек

legacy://

Как правило имеет смысл оставить этот источник данных первым. настройки для него делает tower. Текущая башня не способна сформировать полный список параметров. Опционально можно указать какой файл будет читаться вместо стандартного `/opt/noc/etc/noc.yml` Скорее всего в этом не очень много смысла.

yaml://

Файл с настройками определенными в ручную. Механика скорее всего выглядит так

```
./noc config dump > /opt/noc/etc/settings.yml
```

и далее ручное редактирование полученного файла.

consul://

Для крупного кластера удобнее когда настройки хранятся централизованно. Consul как раз годится на эту роль. формат определения `consul://<ip1>:<port>/<path>?token=<token>`

Рекомендуемый вариант consul://consul:8500/noc.

В дальнейшем настройки читаются из consul при помощи

```
% consul kv get -recurse noc/language
noc/language:ru
noc/language_code:ru
# consul kv put noc/language en
Success! Data written to: noc/language
% consul kv get noc/language
en
```

или через штатный web ui consul, который расположен на хосте с NOC-ом по адресу http://noc:8500/

env://

Переопределять настройки конкретного демона, например loglevel удобнее через переменные окружения. Сейчас используется для переопределения настроек в supervisord.

не исключено появление других вариантов хранилищ конфигурации. пример варианта реализации можно посмотреть в `core/config/proto/yml_proto.py`

Как посмотреть какая конфигурация в конечном счете будет использована сервисами NOC?

- убедиться что сервис запускается через supervisord. сервисы определены в etc/noc_services.yml там нужно посмотреть секцию Environment скорее всего там будет что то не отличимое от

  ```
  environment = LD_PRELOAD="/usr/lib64/libjemalloc.so.1",NOC_USER="noc",NOC_NODE="noc",NOC_CONFIG="
  legacy:///,yaml:///opt/noc/etc/settings.yml,env:///NOC",NOC_ENV="NOC",NOC_LOGLEVEL="info",NOC_DC="DC",
  NOC_ROOT="/opt/noc"
  ```

- Убедится что вы используете порядок чтения настроек идентичный порядку чтения у демона. для применения выполнить

```
export NOC_CONFIG="legacy:///,yaml:///opt/noc/etc/settings.yml,env:///NOC"
```

- запустить команду `./noc config dump`

Вывод команды будет дан в формате совпадающем с читаемым протоколом yml://.

> Формат yml довольно капризно относится к некоторым спецсимволам см. YAML. Для конфигурации это означает, что возможны сложности с паролями начинающимися со спецсимволов *,%,@, #

Сервисы

Данные о связных системах таких как postgresql, mongo и прочие добываются из consul. Существует два типа сервисов

- near – при резолве будет выбран ближайший сервис. под ближайшим понимается сервис пинг до которого меньше. чаще всего это означает локальный сервис.
- обычный – при резолве будет выбран первый попавшийся сервис.

Есть исключение. Если в сервисе будет встречено двоеточие например так вот NOC_NSQD_ADDRESSES=192.168.1.1:4150 резолв происходить не будет. будет взят указанный сервис

Сервисы ускорения (pgbouncer, redis)

С некоторого количества оборудования встает проблема производительности и масштабируемости. Pgbouncer призван решать вопрос с колвом коннектов к postgres. по умолчанию кол-во коннектов к postgres 302 этого достаточно примерно на 250-500 устройств. Дальше колво коннектов придется расширять и после 5000 потребление памяти postgres станет неприлично большим. Pgbouncer позволяет снизить колво реальных коннектов к Postgres, мультиплексируя их на себе.

Redis хранит данные целиком в памяти и решает проблему с откликом из postgres и mongo. Проблема начинает быть заметной примерно на 50 000 устройств.

Что бы использовать pgbouncer и redis надо явно указать их использование. Для этого где-то в цепочке конфигурации надо задать два параметра.

```
pg:
  addresses: "pgbouncer"
cache:
  cache_class: "noc.core.cache.redis.MemcachedRedis"
```

По умолчанию параметры выглядят так

```
pg:
  addresses: "postgres"
cache:
  cache_class: "noc.core.cache.mongo.MongoCache"
```

То есть для postgres обращение будет происходить напрямую в postgres, а вместо redis будет использован cache в mongodb.

> Redis, хотя быстрее кеширования в mongo не предоставляет отказоустойчивой конфигурации.

Proxy

Для некоторый сервисов например tgsender бывает нужно задать прокси сервер. Сделать это можно через тот же settings.yml

```
proxy:
  http_proxy: http://192.168.1.1:3128
  https_proxy: http://192.168.1.1:3128
  ftp_proxy: http://192.168.1.1:3128
```

по умолчанию эти параметры будут взяты из соответствующих переменных окружения и не требуют конфигурации.