Bergamot Monitoring 4.0

https://bergamot-monitoring.org/

What Is Bergamot Monitoring?

- An open distributed monitoring system
- Rich out of the box functionality and flexible configuration
  - Persistent state
  - Performance data
  - Modern, realtime UI, complete REST API
    - Low latency from execution to UI
  - Fine grained access controls
- Distributed by default
  - Load balancing of checks
  - Geographic distribution of checks
  - Scalable scheduling and result processing
  - Designed with scalability and flexibility in mind
    - From 1 standalone node to many nodes

What Is Bergamot Monitoring?

- Plugable check execution and notification engines
  - 8 check engines, 4 notification engines
  - Scripted checks enable Real User Monitoring checks
- Multi-tenanted
  - One cluster
  - Many completely isolated Sites
- Has a migration path from Nagios
  - Can convert existing Nagios configuration
  - Native, non-blocking, efficient NRPE support

A Bit Of History

- I started the project back in 2014/5
    - Original idea came out of writing a Nagios config parser
- The idea evolved quickly and released three versions upto 2018
- Got a bit disillusioned for various reasons
    - I ended up taking a bit of a break
- Started working on it again, revamping it: 4.0.0
    - Simplifying deployment
    - Improving resilience and reliability
    - Rearchitecting core communications
    - Upcoming 4.0 release aims to fix the drawbacks of the earlier releases
        - Significant change from the past
        - And provide a base for going forward
- The name comes from my like of Earl Gray tea, which has Bergamot in it.

Overview

Dashboard

- The dashboard of Bergamot Monitoring gives an overview at a glance
  - Focuses on showing most important information and enabling drill down
    - Active Alerts, Groups and Locations
- The dashboard is realtime
  - As soon as a check result is processed, the new state is pushed to all browsers
  - Alerts will drop off if they recover or are acknowledged
  - Groups and Locations have state just like checks
    - Based on all checks and sub groups/locations
- The dashboard tries to be pretty
  - Icons and various organisation of checks is all configurable

Groups

- Groups hierarchically organise checks
- Checks can be in one or more group
- Groups can be in one or more group
- Have state too
  - Computed from:
    - All child checks
    - All child groups (recursively)
- What Nagios calls *service_groups* and *host_groups*

Locations

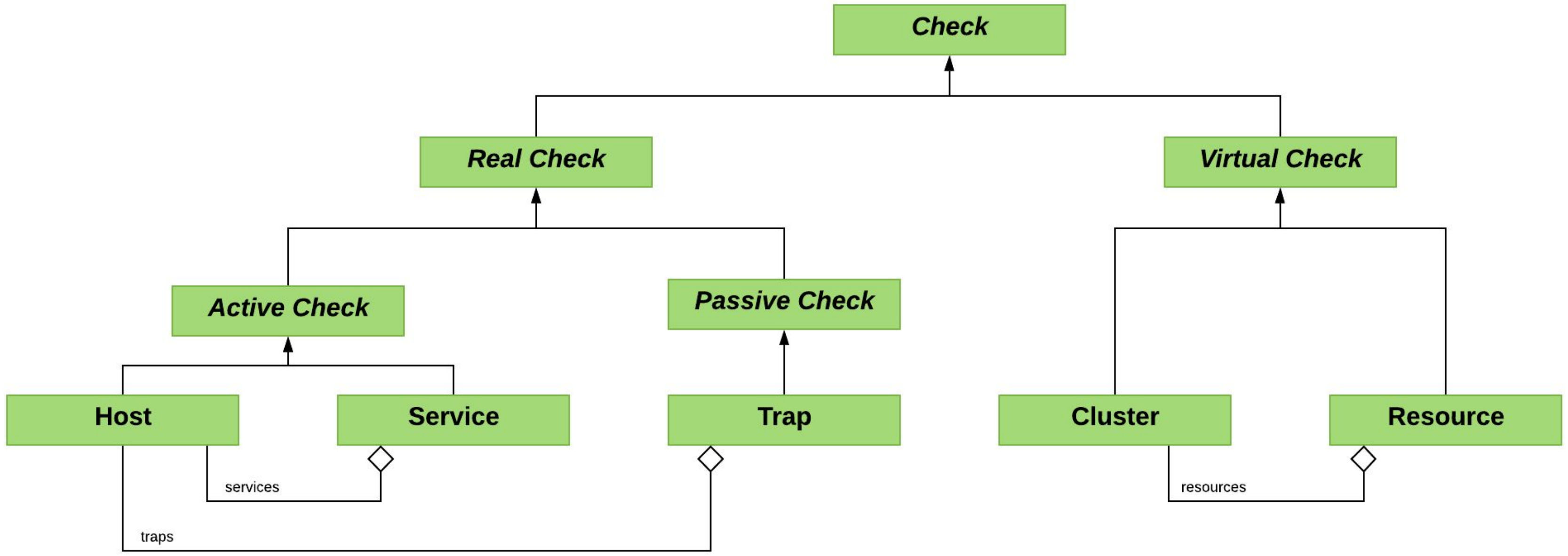
- Locations hierarchically organise checks
- Model physical deployment
        - Main Data Centre, DR Data Centre
- A location can be in one location
        - AWS -> EU West 1
        - UK -> UK Office
- A check can be in one location
- Configuration properties can be attached to locations
    - This can then be used by all checks in that location

Checks

Checks

- Checks are assertions about your infrastructure
- Checks have a state:
  - Ok or ! Ok
  - Status: Ok, Warning, Critical, Info, …
  - Steady or Changing
    - Needs *n* attempts to reach a steady state
  - Human readable output
- Very similar to Nagios and most monitoring systems

Check Model


Check
Real Check
Virtual Check
Active Check
Passive Check
Host
Service
Trap
Cluster
Resource
services
traps
resources

Active Checks

- Active checks are scheduled
  - Have three intervals: steady, changing, retry
  - Constrained by time period
- Active checks send commands which are executed by check engines:
  - Nagios (good old fashioned Nagios plugins)
  - NRPE (NRPE, but without the overhead of check_nrpe)
  - Bergamot Agent (Bergamot Monitoring version of NRPE)
  - HTTP (Make HTTP checks)
  - SSH/SFTP (Run Nagios checks over SSH, login to SFTP servers)
  - JDBC (Query Databases)
  - JMX (Query Java Applications)
  - SNMP (Query SNMP agents)
- Most (native) check engines support scripting

Hosts

- Hosts are active checks which aim model servers or devices
- Hosts contain Services and Traps
- Pretty much the same concept in Nagios

Services

- Service are checks about or on a Host
- Pretty much the same concept as Nagios

Passive Checks

- Passive checks are not scheduled
- When a result is receive the check state is computed
- Results can be matched by various means to a check
  - UUID
  - Name
  - External References
  - ...

Trap

- A Passive version of a Service
- Same concept as a Service with a passive command in Nagios

Virtual Checks

- Behave like checks
  - Have state
  - Can raise alerts
- However they reference real checks
- Computed when dependent checks change
- Designed to model multi-node clusters
  - Where your cluster is health if one or mode node is
- IIRC similar to check_multi plugin in Nagios
  - but in real time, as things happen

Cluster

- The virtual version of a Host
- Contains zero or more Resources

Resource

- The virtual version of a Service or Trap

Metrics

Metrics

- Out of the box Bergamot Monitoring supports metrics
  - Any Nagios plugin performance data is captured and stored
  - Metric data is displayed with checks
- Native check engines also support and emit metrics
- The metrics are stored in PostgreSQL
- Currently can't be used for more than just looking at

Alerting

Alerts

- When checks fail, alerts are raised
    - Alert history is stored and kept
- Alerts will recover (hopefully)
- Alerts can be acknowledged
    - Alerts can escalate if not acknowledged or recovered within time windows
- Notifications may also be sent to Contacts
    - Filtered at both check and contact level
    - Notifications are sent via notification engines:
        - Email
        - SMS
        - Slack
        - Webhook

Downtime and Dependencies

- Downtime can be added to checks to avoid alerts during maintenance
  - Downtime can be added via the UI or API
  - Currently scheduled downtime isn't supported
- Checks can also Depend upon other checks
  - Same as parenting in Nagios
  - Services on a Host automatically depend on that Host
    - If the Host alerts, all Service alerts are encompassed to that alert
  - Alerts won't be raised when a dependent check is in downtime

SLAs and Status

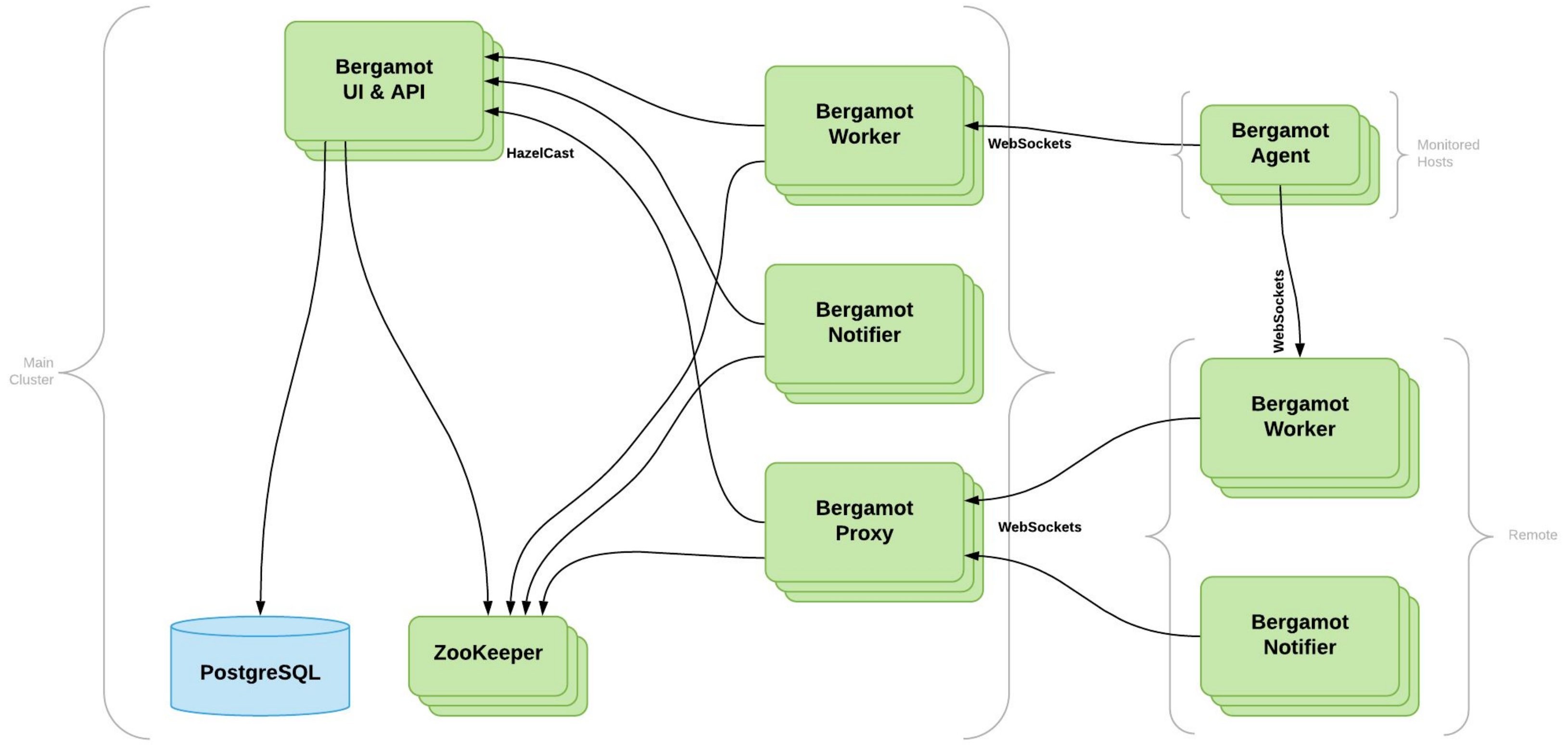
- Alert history is persisted allowing for SLAs to be computed
    - SLA reports can be configured on checks
    - Alerts can be marked as a false positive to be removed from SLA calculation
- Checks can optionally be displayed on a public Status page
    - This allows high level service status pages to be public
    - SLAs can optionally be displayed on the status page

Distributed

Distributed

- Bergamot Monitoring is distributed by default
  - The core scheduling, result processing and UI components can cluster
    - Can be run standalone on one server for simple setups
  - Workers and Notifiers are also distributed
    - Useful even if not needed for scaling
- Workers are distributed for:
  - Load balancing, just run the checks over a bunch of machines
  - Geographically, pools of workers for a specific task or location
- Decoupled Workers and Notifiers enable extensibility
  - Custom workers and notifiers can be created and can connect in
    - Just requires WebSockets and JSON to get going
  - Misbehaving check commands have limited impact on the core cluster

Distributed



Geographic Distribution

- Workers can be tagged into pools
    - A worker can be configured with its worker pool on start up
    - Worker and worker pools are dynamic, they can come and go with no effort need centrally
    - Checks are load balanced over workers within the same pool
- Checks can be assigned a worker pool to use
    - For Hosts this flows from its Location
    - For Services this flows from its Host
- Checks assigned to a worker pool will only be executed those workers
    - If there are no workers in a pool, the check will fail as `unroutable`

Configuration

Configuration

- Checks are configured using an XML dialect
  - Designed to be compact and readable
- Configuration model is based on inheritance
  - Check configuration extends template configuration
    - Can inherit from multiple parents many times
- Idea is to configure something once, and reuse it
  - Hosts and Cluster inherit Services and Resources
  - Reconfigurable a template will cascade to all changed checks
- The check model is also used to an advantage
  - Configuration for a Host can be found via the Location it is in
- Configuration changes happen live, without any restarts
  - Configuration changes applied as atomic units
  - Audit history of configuration changes is stored

Configuration

- As well as checks:
  - Contacts - People to notify and login to the UI
  - Teams - Groups of people
  - Credentials - Storage of credentials needed by checks
  - TimePeriods - Time ranges used for scheduling and notifications
  - Commands - The actual check that gets executed

Configuration

- Commands are the shire horse of active checks
  - They define what will be executed by a worker
  - They are essentially:
    - The check engine and executor name
    - And a bag of name, value pairs
    - Parameter values can also contain expressions
- A lot of engines support scripting
    - This allows complex checks to be created as pure configuration
    - Rather than dedicated plugins needing to be written and deployed to workers

Agents

Agents

- Bergamot Agent is a service than can be run on monitored hosts
  - Provides core OS level monitoring out of the box
  - Supports being able to run Nagios plugins
- Agents can autoregister on connect
  - At first connect the Host configuration is created based on a given template name
- Agent Keys
  - Agents are authenticated by shared keys
  - Can be generated from the UI or API

Security

Security

- Bergamot Monitoring has a fine grained access control system
  - Permissions can be granted or revoked to every check for every user
- To make this practical
  - Checks are placed into Security Domains
    - Inherited as normal via configuration templates
  - Contacts are granted permissions over Security Domains
  - Contacts inherit permissions of the Teams they are in
- The can build very powerful setups
  - Only DBAs can see, configure, acknowledge database checks
  - Configuration change work flow can be enforced
    - Contacts can create but not apply configuration changes
    - Contacts can only see their configuration
- 2FA is supported and can be enabled, allowing login with YubiKeys

Tenants

- Bergamot Monitoring also supports multiple tenants
- One cluster can run many Sites
  - Sites are based on the virtual host name (URL)
- Sites are isolated from each other, having their own configuration and URL
  - Contacts, access controls, etc are all part of configuration and isolated
    - The same email address can exist in multiple sites
- First contact of the first site is a Global Admin
  - Can see cluster status, manage sites
  - Can add other global admins
- Global Workers and Notifiers are shared
  - But sites may have their own dedicated Workers
  - Care does need to be taken in naming global worker pools

Migrating

Migrating

- A number of design decisions exist to enable a migration path from Nagios
- Bergamot isn't a drop in replacement, but is meant to be low effort
- Nagios configuration can be converted
  - CLI tool will read Nagios configuration and output Bergamot configuration
  - The convert tries to be smart where it can
    - Computing and outputting template configuration
  - Converted configuration is valid, but might want to be manually tweaked
- There are a few things that can't be easily migrated
  - Singleshot or very short scheduling period checks will need to be changed

Future

Future

- Evolving the check model
  - Top level Services
  - Modeling of containers / applications
- Supporting more Metrics based approaches
  - A lot of modern monitoring setups now just rely on metrics
  - Aim to get Bergamot Monitoring to be a hybrid of both worlds
- Improve downtime UI
- Improve config editing UI
- Cloud based service

Thanks For Listening