Monitoring Complexity

Open Source Distributed Monitoring

Chris Ellis - @intrbiz - @bergamotmonitor

Setting The Scene

- IT is getting increasingly complex
  - Multiple physical and virtual sites
  - Wide range of application technologies

- IT is essential to our daily lives
  - Loss of business due to services being unavailable
  - Availability is now critical

What Is Monitoring

- Checking your infrastructure and systems for problems, constantly, 24x7, 365 days a year

- Taking action as soon as something has gone wrong, usually waking somebody up

A tree falls in the forest.
Nobody is there to hear it....

Does it make a sound?

Why Do We Need Monitoring

- Some issues are easy to spot
  - It is 02:00, you are tucked up in bed, your website has just gone down
  - Your customers spot the problem
  - You wake up to angry complaints

- Monitoring helps you spot issues before someone important does.

Why Do We Need Monitoring

- Some issues don’t immediately surface
  - Your database slave falls behind or fails
  - This doesn’t immediately impact your systems
  - Everything keeps running, nobody notices the issue
  - Main database server crashes
  - Suddenly, you have no slave, or an out of date slave to replace it
- Monitoring helps your spot non-obvious issues

Bergamot Monitoring

Open Source Distributed Monitoring

Bergamot Monitoring

- Started by me, just over a year ago, version 1 released, version 2 coming very soon

- Not far off my 1,000 commit
- ~79,000 lines of code in Bergamot
- ~60,000 lines of code in supporting modules

Features - Nagios Compatible

- Executes Nagios plugins, reuse any existing Nagios plugin you are using or can find
- Native, non-blocking NRPE support
- Import and convert pre-existing Nagios configuration
- Easy migration path from Nagios

Features - Distributed

- Distributed by default, not added as an afterthought
- Allows for geographic distribution of check execution
- Only need to distribute workers node geographically

Features - Modular

- Workers
  - Nagios / NRPE
  - SNMP
  - HTTP
  - Bergamot Agent
  - TCP / UDP *
  - JMX *
  - JDBC *
  - LibVirt *
- Notifiers
  - Email
  - SMS
  - WebHook
  - IRC *

Features - Scalable

- Check executions naturally load balance across available workers
- Easy to add more check capacity by deploying new workers
- UI clustering allowing for a horizontal scale out. Spreading UI, scheduling and result processing load across a pool of servers

Features - Resilient, Secure

- Can tolerate the loss of worker, notifier and UI nodes.
- Should a node fail, another picks up the work
- Should all workers fail, checks will timeout and alert
- TLS used by default, Certificates used to secure monitoring agents

Features - Persistent

- Configuration, state, metrics - are all stored in a PostgreSQL database
- Possible to write complex reports against monitoring state using external tools
- Configuration changes are made online, requiring zero downtime and no reload windows to apply

Features - Modern UI, Real Time

- A modern user interface, offering a clean, visual presentation of monitoring state
- Real time updates via WebSockets. Checks can execute and be displayed in a matter of milliseconds
- Real time alerts to the browser should a check alert

Features - Multi-Tenanted

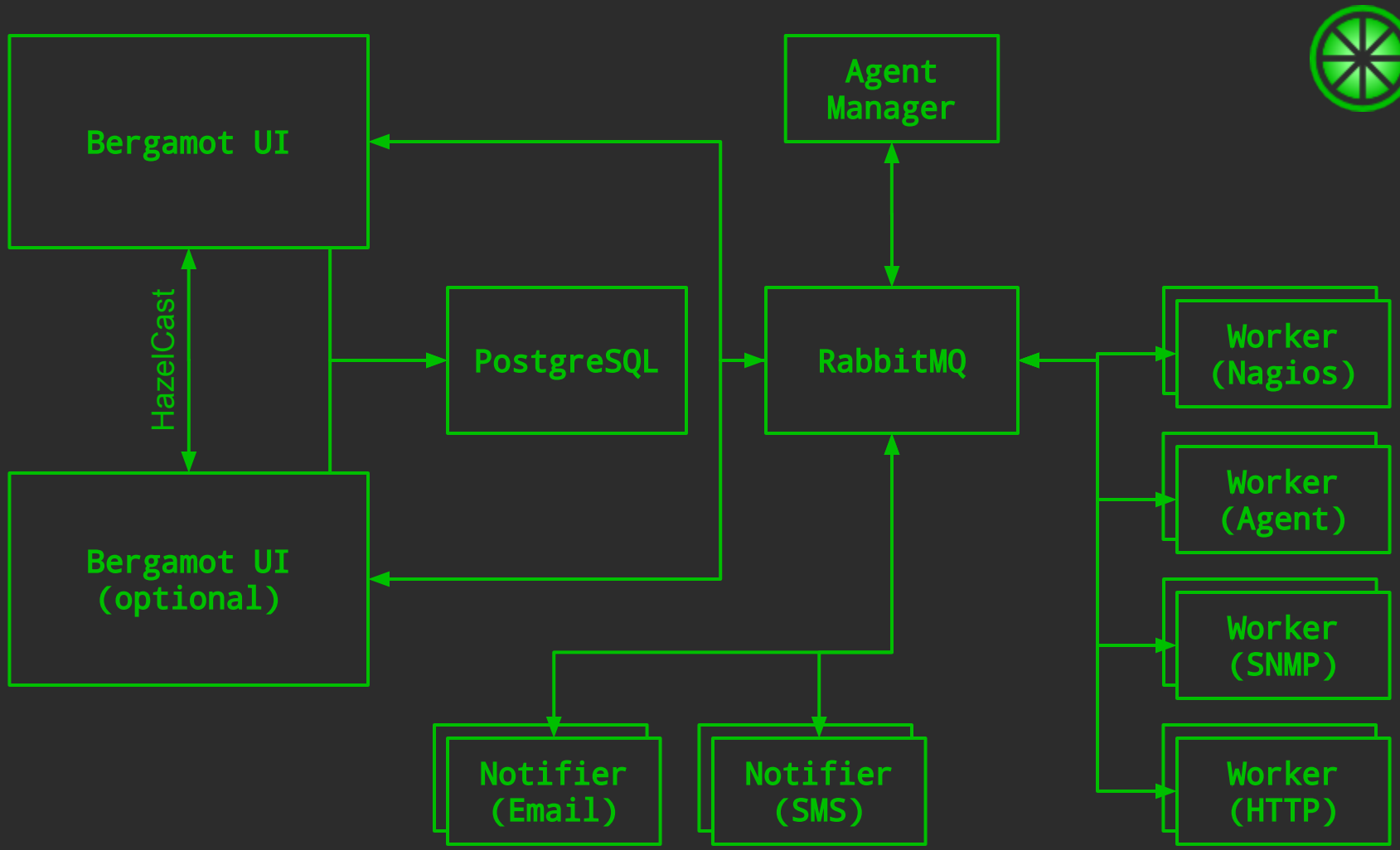
- Designed from the ground up to be able to run multiple, isolated monitoring systems from the same infrastructure
- Just like virtual hosts in web servers, Bergamot Monitoring can run many virtual sites from a single installation

Features - Extensible, Open

- A JSON REST API allows for people to integrate and extend Bergamot Monitoring easily.
- Scriptable check engines, fast, efficient checks implemented using JavaScript
- Easy develop new check engines
- Open Source - Under LGPL V3

Architecture

- Functionality is separated into different services, which communicate via message queues
- Each service is specialised, doing one thing and doing it well
- Services are loosely coupled, making it easy to modularly add new functionality


Bergamot UI
Agent Manager
HazelCast
PostgreSQL
RabbitMQ
Worker (Nagios)
Worker (Agent)
Worker (SNMP)
Worker (HTTP)
Bergamot UI (optional)
Notifier (Email)
Notifier (SMS)

Architecture

- Workers
  - Handle actually checking things, specialised by what they check, eg: nagios, NRPE, SNMP, HTTP
- Notifiers
  - Alerts people and robots when the shit hits the fan
- UI
  - Runs the UI
  - Scheduling and result processing
  - Clustered for HA

Object Model - Checks

- Active
  - Hosts - a device
  - Services - something on a device
- Passive
  - Traps - something on a device
- Virtual
  - Clusters - a cluster of devices
  - Resources - a cluster of services

Object Model - Organisational

- Groups - arbitrary grouping of checks
- Locations - physical location of a host

- Categories - groups of checks
- Applications - grouping of application checks

Object Model - Misc

- Commands - check definitions
- Time Periods - when stuff can happen

- Contacts - people to be notified
- Teams - groups of people

Configuration

- XML based configuration, designed to be readable and compact
- Uses inheritance heavily, define core configuration once and inherit it
  - Hosts inherit Services and Traps from parent
- Aims to make templates easy, define a template once, use it for many times

Bergamot Agent

- A host agent, like NRPE, executes checks on the actual server, eg: CPU usage
- Checks are not configured on the server, the agent is told what to do, less overhead
- A lot out of the box
  - CPU, Memory, Disk usage
  - Processes, Network Connections
  - Network and Disk IO

Bergamot Agent

- Heavily secured
  - Connects out using WebSocket to central hub
  - TLS certificates secure the server and the cluster
  - Each host identifier is in the certificate and signed
- Bergamot Agent Manager handles signing
  - As a separate service for security reasons
  - Can be stored on an isolated machine
  - Generates per site keys and certificates