ZTF Alert Distribution System
(ZADS Operations)

Mario Juric, Maria Patterson, Eric Bellm

ZTF Alert Stream Readiness Review

May 4, 2018

Outline: ZTF Alert Distribution to Public Brokers

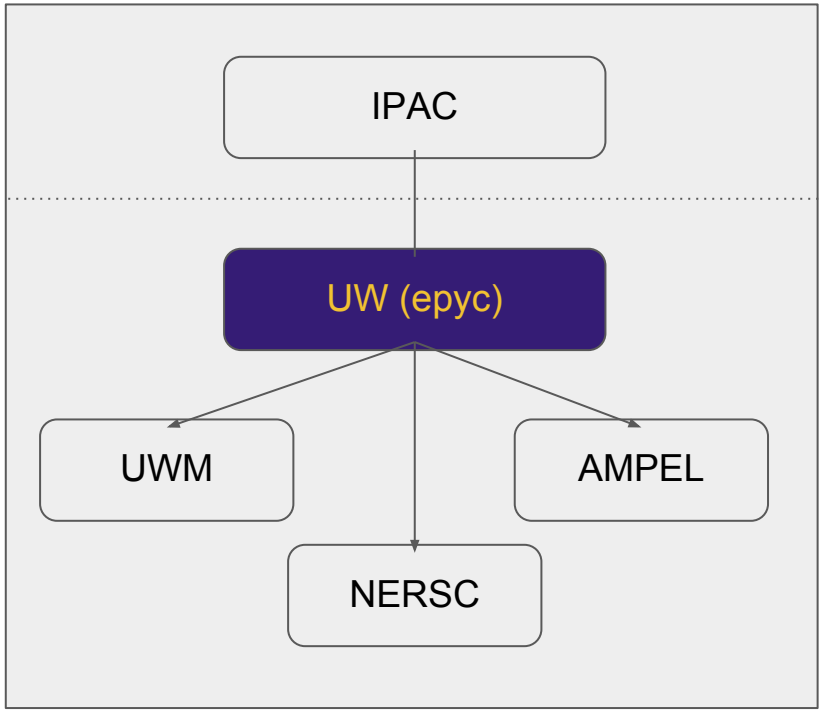
- Goals
- Architecture
- Performance
- Reliability
- Monitoring and Alerting
- Cybersecurity
- Deployment, Backups, and Disaster Recovery
- Documentation
- Next steps

Partnership Alert Distribution System

For the past ~3 months, we've been operating an alert distribution service for the ZTF partnership.

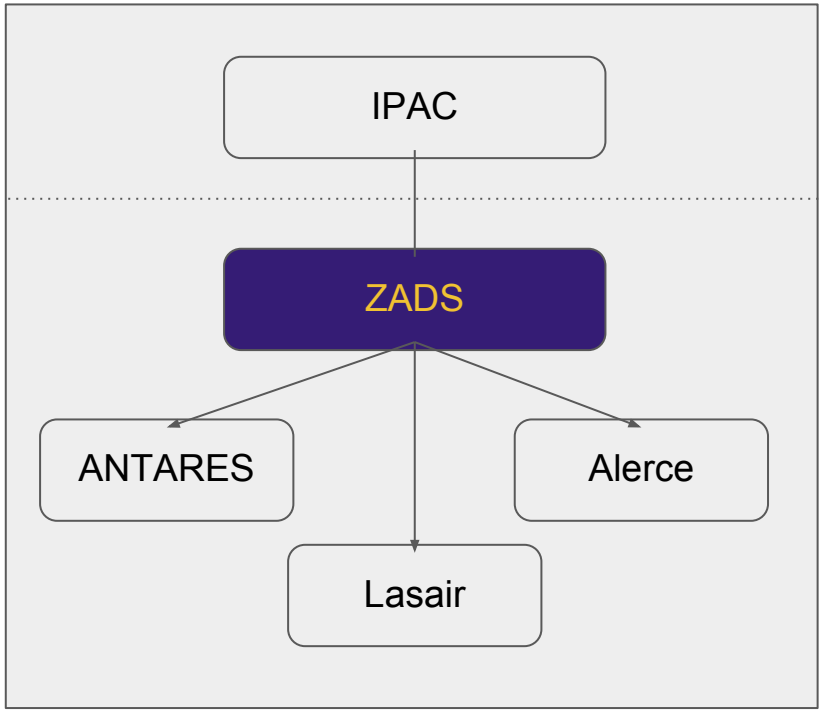
The service runs at a machine at UW (epyc.astro.washington.edu), receives alerts from IPAC, and forwards them to partnership event consumers.

We're looking to incorporate lessons learned from running this service to deploy a reliable system that will provide the same service to public brokers.



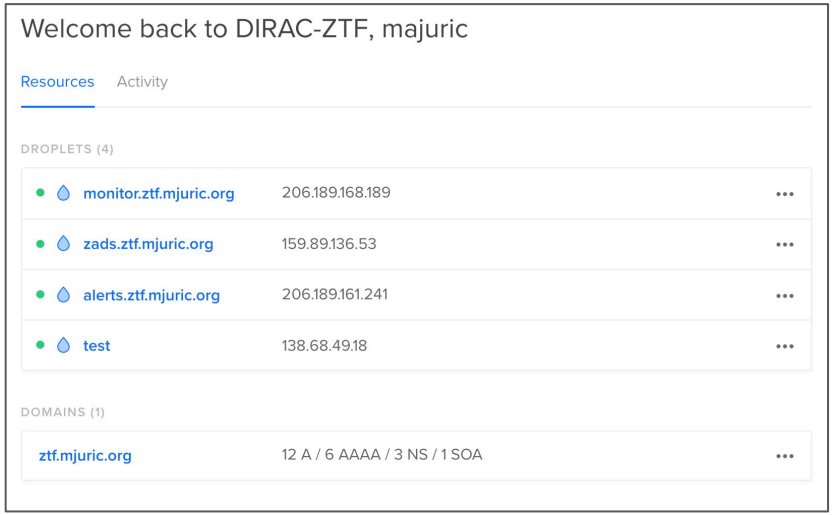
Goals for ZADS Public Operations

- Receive the alert stream from IPAC and forward it, without filtering, to a small number of public alert brokers. The general public will access the alert stream via the brokers.
  - Aim to initially support N<~5 brokers, with focus on ANTARES, Lasair (UK), Alerce (Chile)
- Sufficient throughput for real-time operation
  - Up to 1 MBytes/second/broker, sustained.
  - Aim for >=5x burst speeds.
- 99% reliability (1 full night/quarter downtime)
  - Allowed for shorter downtimes, e.g. occasional 10-15 minute windows.

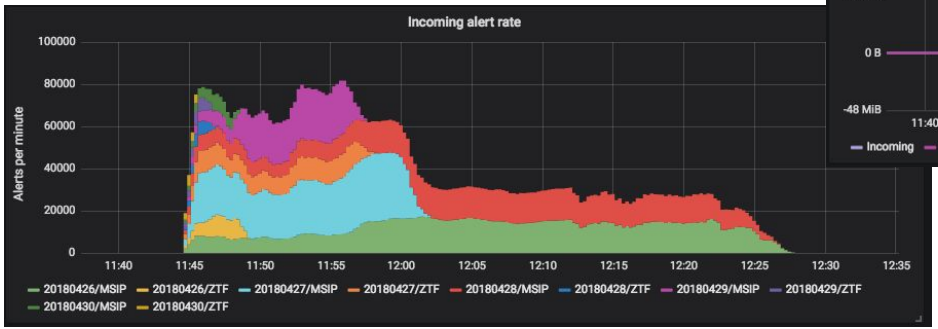


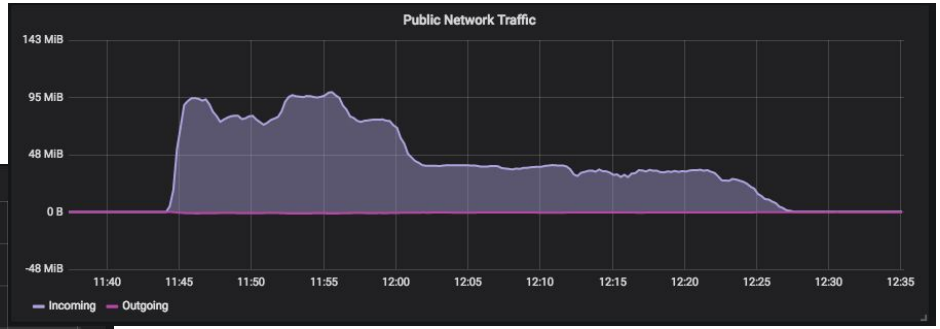
Architecture

- Cloud-hosted solution on Digital Ocean (DO)
  - Dedicated machine(s), decouple from other work on epyc (shared machine)
  - Deployed in San Francisco data center
- Elements
  - **Broker (zads.ztf.mjuric.org)**
    - Single VM (6 cores, 8 GB RAM, 320 GB local SSD)
    - Runs a single instance of Zookeeper, Kafka, and MirrorMaker
  - **Monitoring (monitor.ztf.mjuric.org)**
    - Single VM (1 core, 512 GB RAM).
    - Runs Prometheus and Grafana



Performance





- **Incoming:** We're seeing **40-90 Mbytes/sec** ingest rates when mirroring from UW to DO. We expect similar rates from IPAC. ✓
- **Outgoing:** We've tested consuming with kafkacat, seeing **~5 Mbytes/sec** (independent of the number consumers, measured up to 5). ✓
- **Notes:** DO does not guarantee bandwidth. We've observed ***significant*** variance from VM to VM as to the realized bandwidth. The quoted numbers are for the ***worst*** VM (with the best VMs we achieved 90MBytes/sec, outgoing).

Reliability I/II

- **The system as-built has single points of failure**
  a. **Any of the components may go down (zookeeper, kafka, mirrormaker)**
     - **Assessment:** Running on epyc for ~two months we've observed these rarely (if at all). We've set up systemd to automatically restart in case any component goes down.
  b. **The host may go down (or be taken offline for maintenance)**
     - **Assessment:** We have experience with DO hosts -- over the past 4 years, we've observed a single instance of downtime lasting longer than 6 hours.
  c. **DO's datacenter may get cut-off from the internet**
     - **Assessment:** See above.
  d. **Human & management instrumentation error**
     - **Assessment:** We utilize OS-es and deployment technologies we're familiar with (CentOS 7, systemd).

- **Overall: detect issues in real-time, attempt to fix automatically, alert the operators otherwise.**

Reliability II/II

- **We've examined (and tried) alternative architectures:**
  a. ***Three kafka cluster nodes in the same DO datacenter, with replication factor of 2***
     - Defends against a) and b) above, but...
     - … significantly complicates management (e.g., service startup/teardown needs to be synchronized, other issues). The management complexity increases the crossection for channel d), making it less reliable than a single-host solution.
     - This is the right solution in the long-run, but will require deploying something like Kubernetes.
  b. ***Three geographically distributed kafka cluster nodes (SFO2, TOR, NYC3), replication factor of 2***
     - Would defend against a), b), and c), but…
     - The increased cross-datacenter latency made reliable kafka replication difficult. Kafka is not generally meant to be used in a geographically distributed cluster. Still, it's an interesting solution to further explore in the long-run.
- **Bottom line: to meet the initial goal of 99% uptime, we've judged the less complex and more familiar architecture as the one with less overall risk.**

Monitoring and Alerting

- We deployed **Prometheus** as our monitoring solution
  - Collect hundreds of host and application metrics that can be used to monitor and alert on the health of the system
- We deployed **Grafana** as our visualization and alerting solution
  - Allows the creation of dashboards visualizing system health and status
    - Have one on permanent display at UW DIRAC big screen
  - Allows the creation of alerts (i.e., host down, service down, alerts appear to come in too slowly)
  - In progress: connecting alerts to paging systems (e-mail, Slack, PagerDuty)
- Both services run on a separate DO host (same data center)

Zwicky Transient Facility Alert Distribution System Monitor (monitor.ztf.mjuric.org)

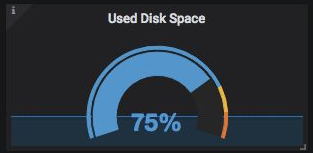
ZADS Health

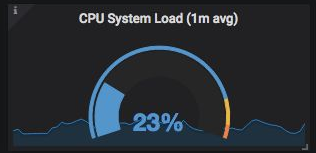
MirrorMaker Uptime: 1.1 days

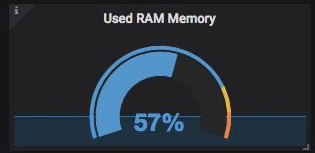
Kafka Uptime: 1.1 days

Zookeeper Uptime: 1.1 days

Host Uptime: 2.9 days







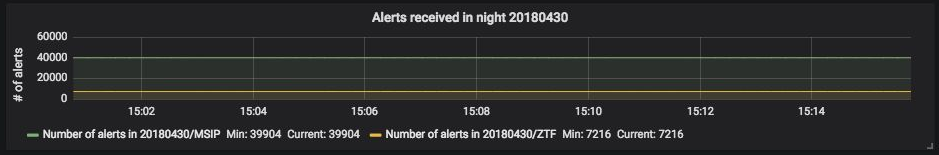
ZADS Report

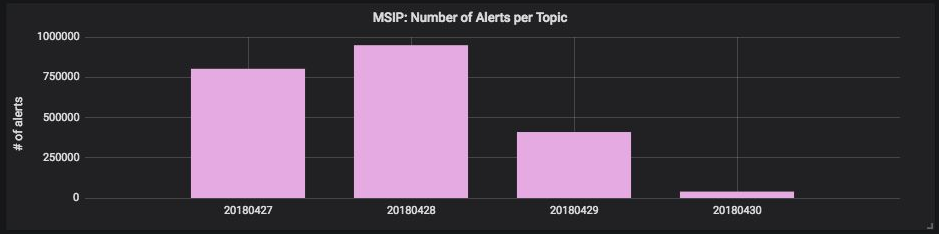
MSIP: alerts in 20180430

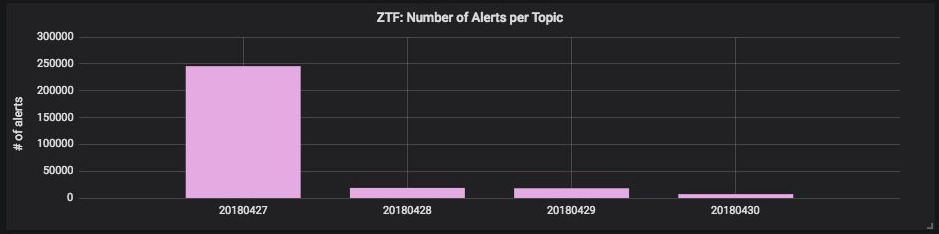
39904 alerts

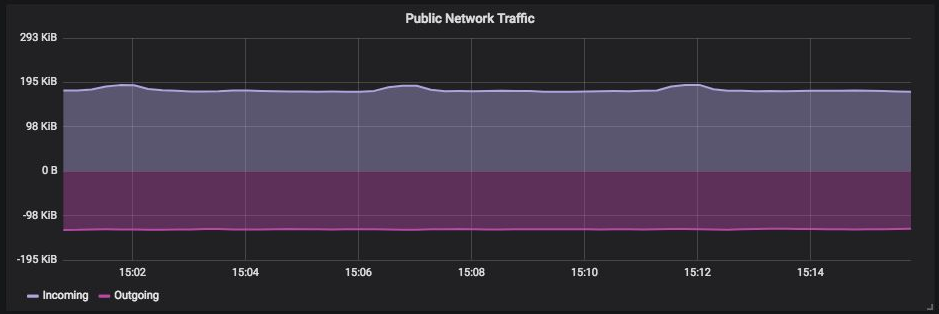
ZTF: alerts in 20180430

7216 alerts









BytesIn Rate

BytesOut Rate

Node Exporter Full

Last 5 minutes Refresh every 5s

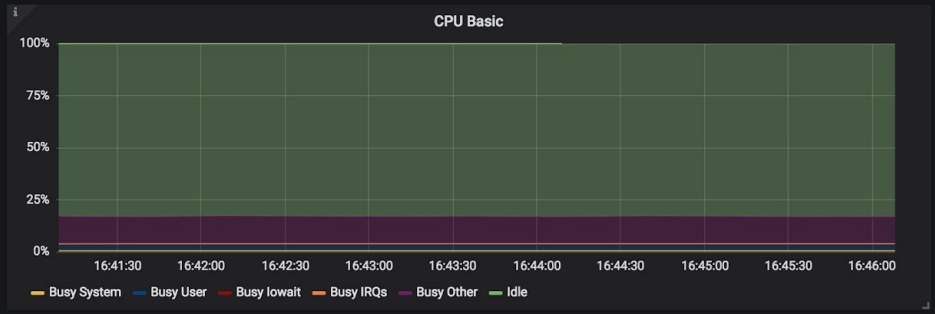
Job: kafka-node | Host: priv-zads.ztf.mjuric.org | Port: 9100

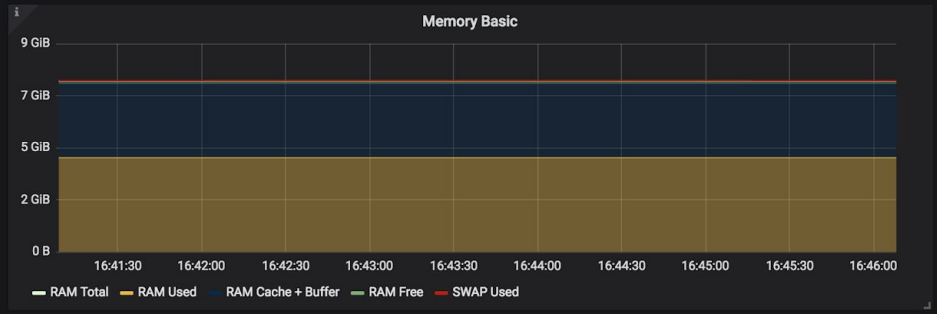
Basic CPU / Mem / Disk Gauge

Basic CPU / Mem / Disk Info

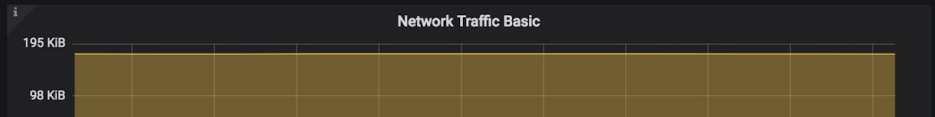
CPU Cores	Total RAM	Total SWAP	Total RootFS	System Load (1m avg)	Uptime
4	7.64 GiB	8.00 GiB	160 GiB	0.68	2.9 days

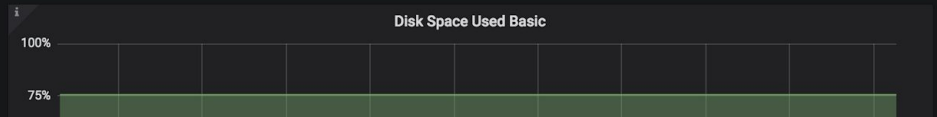
Basic CPU / Mem Graph





Basic Net / Disk Info





Cybersecurity

- **Host security: allow only what's necessary and explicitly permitted, deny rest.**
  - Separate, private, network for zads <-> monitor communication
  - Firewalls up (managed with firewalld), all ports blocked by default
  - SELinux active (enforcing mode)
  - Services not accessible from the outside bind to localhost (or private network interface)
- **Monitor security (monitor.ztf.mjuric.org)**
  - Monitor system uses username/password authentication
  - SSL encryption (using Let's Encrypt)
- **Broker security (zads.ztf.mjuric.org)**
  - Authentication and authorization is still TBD for broker connections. For now, A&A are done on a per-IP basis (i.e., we'd allow only the ANTARES IP to access the broker).
  - Encryption is still TBD.
- **Both systems otherwise isolated from upstream and downstream, mitigating the impact of a potential security breach.**
- **Package update policy: update by hand on a regular basis (details TBD).**

Deployment, Backups, Disaster Recovery

- Given these are “bridge” machines between IPAC and public brokers, carrying little state of their own, the impact of a complete loss is largely in incurred downtime. ***We therefore primarily concentrate on quickly re-creating the service, rather than continuous backup.*** Examples of (worst-case) data loss:
  - Loss of stored offsets for the given week (some clients would see repeated alert packets)
  - Loss of stored monitoring information.
- **All machines deployed to DO using *cloud-init* scripts**
  - It’s possible* to re-create the existing system automatically, within minutes
  - Scripts maintained in a github repository
- **Weekly backups enabled on both machines**
  - These are DO-provided as a service, with copies kept in the same data center

Documentation

- Core need is to support small number of community brokers
- Existing stream listener documentation (Github READMEs) has enabled usage within the collaboration; would like to improve rendering
- Useful standalone documentation for alert packet contents exists

ztf.alert https://zwickytransientfacility.github.io/ztf-avro-alert/schema.html

The top-level alert contains the following fields:

Field	Type	Contents
`objectId`	long	unique identifier for this object
`candid`	long	unique identifier for the subtraction candidate
`candidate`	`ztf.alert.candidate`	candidate record
`prv_candidates`	array of `ztf.alert.prv_candidate` or null	candidate records for 30 days' past history

- *Need clear user-level data access information (and expectation-setting) at ztf.caltech.edu*

Next steps

- **Begin testing with destination brokers (ANTARES) and source sites (IPAC)**
  - The final choice of VM, and the final parameters of the system, should be tweaked to maximize ANTARES <-> IPAC end-to-end *latency.*

- Clean up documentation, complete work on alerting
- Improve monitoring: track and alert on *useful* metrics

- Establish clear procedures, responsibilities, and lines of communication for exceptional situations (e.g., broker down, data transfer issues, cybersecurity issues, etc.)