Ansible Tower with Prometheus

Experiences using ansible and prometheus in your CI/CD workflow

About

56K CLOUND

This talk is about how we built a continuous deployment system for Deep-Impact.

- Spectra is a web application
- Multi-tenant
- Built with Clojure, Mango, Grape, MongoDB, ElasticSearch
- Runs on, AWS: EC2, Container services

[!] DEEPIMPACT

Darragh Grealish
Site Reliability Engineer at 56K.Cloud
DevOps in a Cloud Neutral way
dg@56k.cloud
https://www.56K.cloud/

What we do:

- ➔ Ansible, Docker Engineering
- ➔ CI/CD Improvements and Setup
- ➔ DevOps Bootstrapping
- ➔ CDN Consulting
- ➔ Wireless and Microwave Networking

Background

What triggered the need:

Deployments and Infrastructure managed by a one-click deployment service with Skyliner.io

In spring they announce to shut down in July 15th

- Solely based on AWS
- Deploys one branch/repo per project
- Each project had a QA and Production setup
- Heavy use of EC2, Docker, ALB/ELB and cloudformation

One-click deploys (basically)

skyliner < Skyliner

Dan

/applications/skyliner

precog
US-EAST-1

skyliner
US-EAST-1

Status

Logs

Tasks

Settings

+ NEW

HELP

Build 7e93b1f

Status

Docker image ready
Found an image for build 7e93b1f

The Travis CI build passed
Status updated at 2016-08-23 18:24:03 UTC

Review diff

Deploy to QA

Deploy diff

don't show non-admin users the install region tip
Dan McKinley

removing console.log
Dan McKinley

showing a tooltip when skyliner isn't installed in a region
Dan McKinley

highlighting the new app button when the tooltip is shown
Dan McKinley

showing a tooltip calling out the new app button if there are no apps and a region is installed
Dan McKinley

What is Prometheus

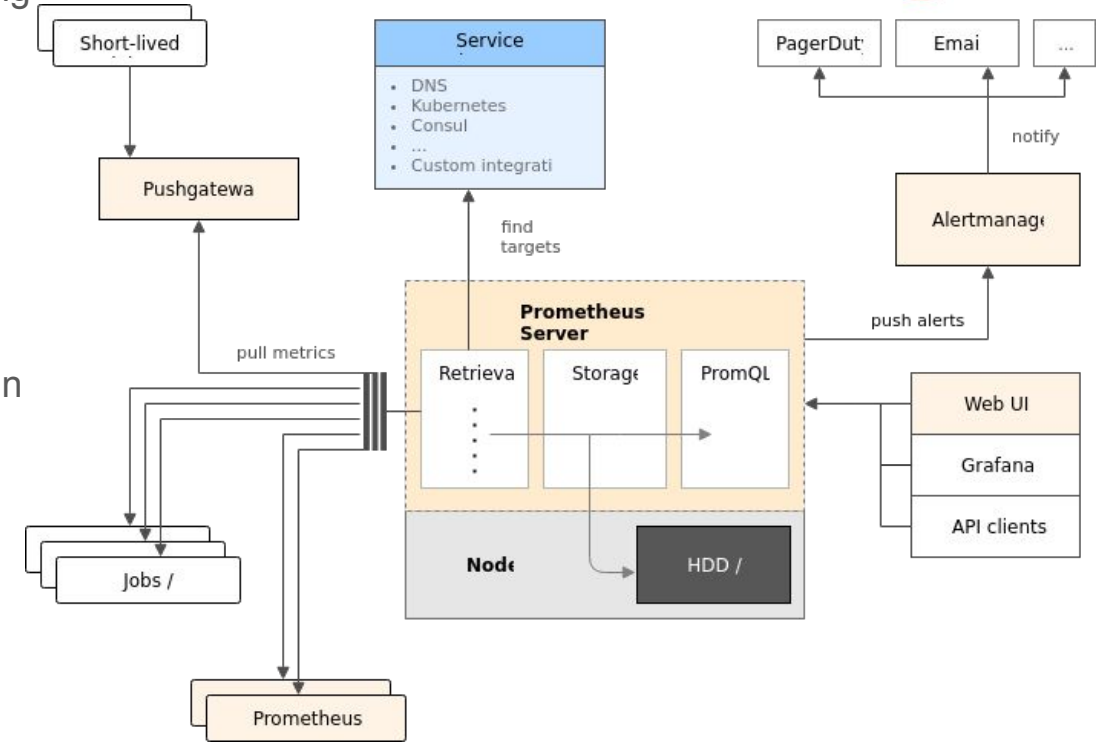
A open-source time-series based monitoring system, founded in 2012 by ex-Googlers working for Soundcloud

Inspired by Google's Borgman internal monitoring tool

Uses time-series data as a data source for generating alerts, this makes it different than other tools out there

Components:

- Prometheus
- Alert Manager
- Node Exporter (agent)



What is Prometheus

Growing bigger,

Github Project:
Almost 4k commits, 11k stars, 1,258 forks,
(178 issues, PR: 45 open / 1632 closed)

16,000 active installations of prometheus running behind Grafana, that report stats

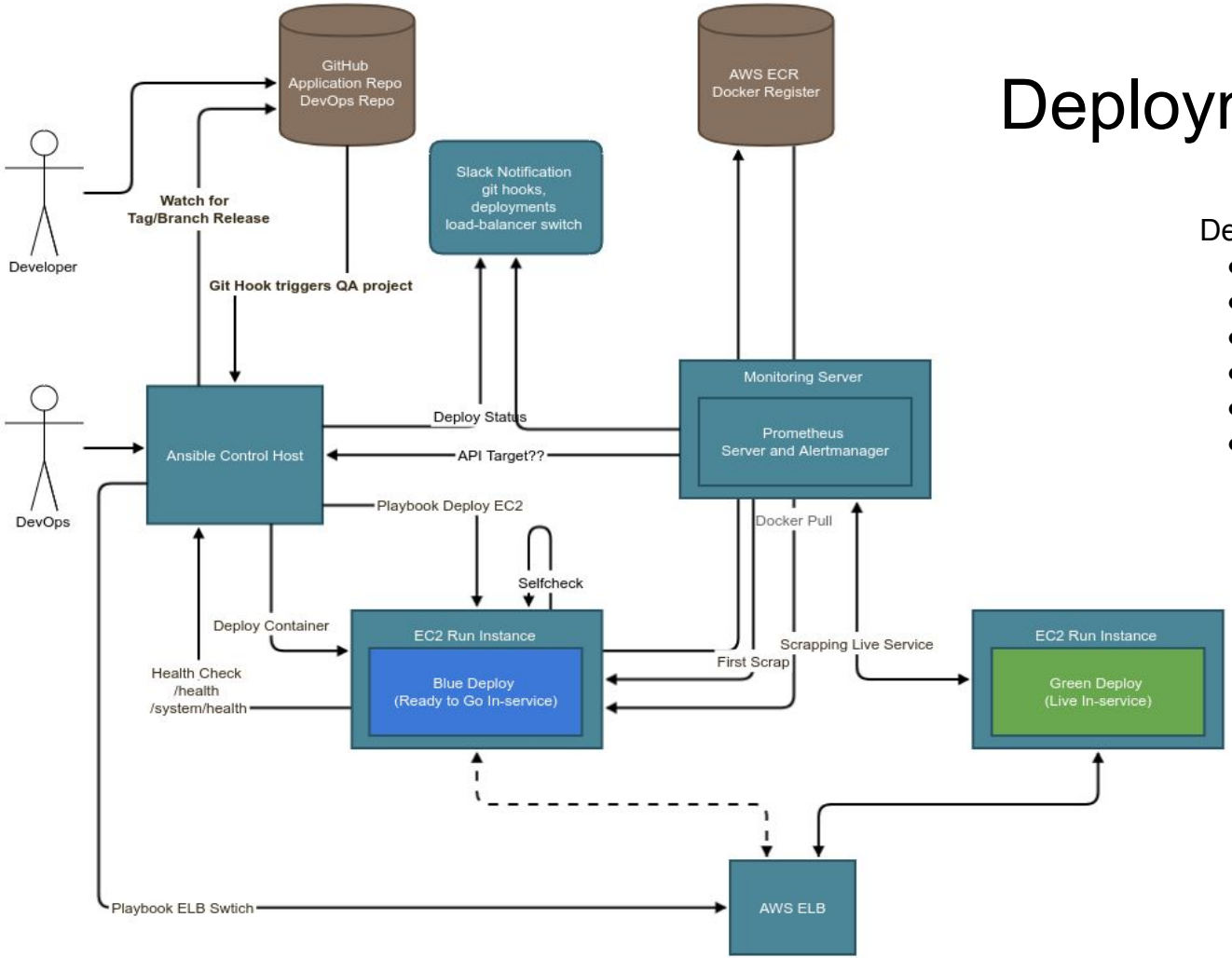
Grafana Talk, PromCon 2017 Munich

Ansible Tower

A nice UI to see all your infrastructure with access control

- Schedule playbook, as Jobs
- Push Button Deploys*
- Dynamic Inventory from AWS
- Free Tier, up to 10 nodes (or inventory items)
- Using a pull model from source control, e.g Github
- Notifications: email, slack
- Access Control, with various integration, Google Auth, LDAP*

Deployment Workflow



Deployment Stack:

- Ansible (/w tower)
- Prometheus
- Docker
- AWS EC2
- AWS ECR (Register)
- AWS ELB (Load-balancer)

Demo

- We will add a new host to EC2
- Ansible Tower will add the host to inventory
- Scheduled playbook will add the node_exporter to the new host and a prometheus target
- Monitoring will start
- An alert will fire when we kill the node_exporter

Links:

http://192.168.106.27:3000/dashboard/db/node-exporter-full?orgId=1&from=now-30m&to=now&refresh=1m&var-node=52.209.114.9&var-port=9100

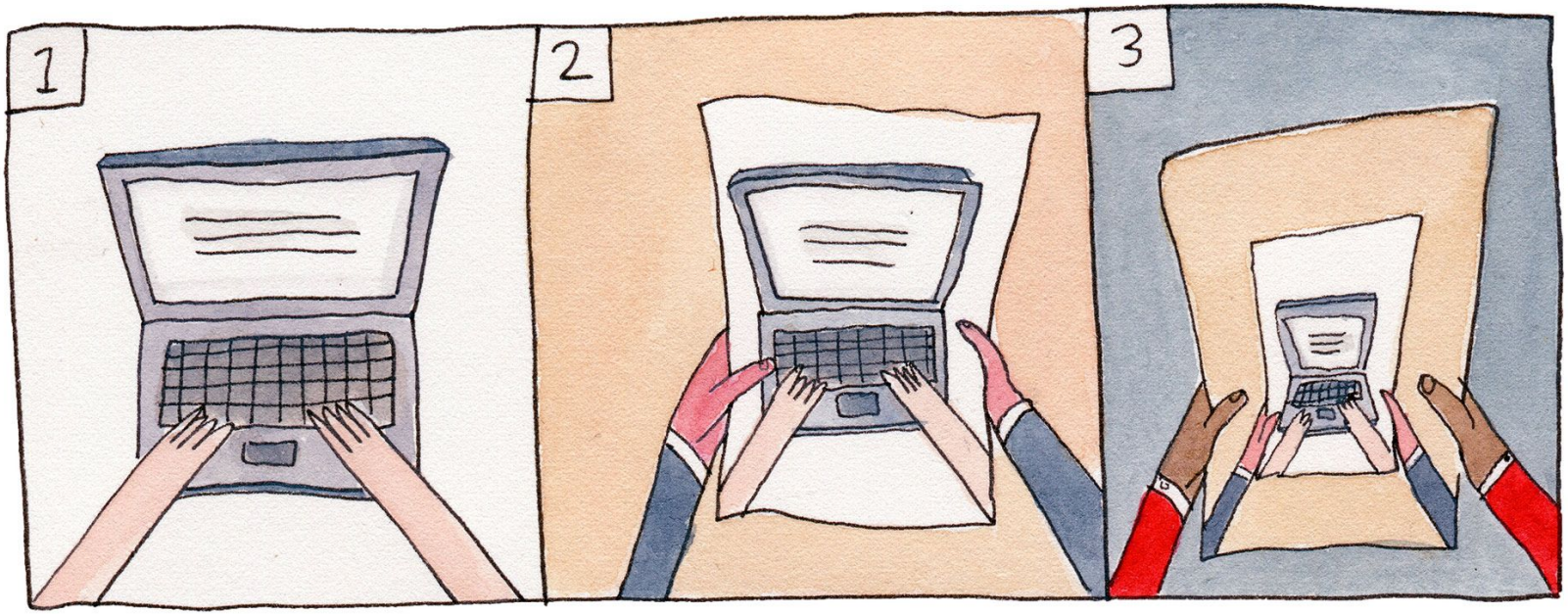
https://192.168.106.15/api/v1/inventories/70/

http://192.168.106.27:9093/#/alerts

Cloud Guy, Trolls Movie, DreamWorks

Looking back ...

3 ~ 6 months on



Source: blog.intercom.com

Looking back… Findings:

3~6 months on:

- One-Click deploys means less flexibility, but robustness
- Unattended automation isn’t easy,
  - Human intervention almost always required to address what state the infra is in
  - Possible mangement want the go-live decision
- Automating to reduce hosting costs , will create
- If your application can’t scale, your automation becomes more complex
  - Deterministic decisions
- Alerting on actionable data require you to gain application knowledge
- Graphing really helps, it visualizes the concern

Tooling: Ansible Tower Pros/Cons

Pros:

- Simple UI for non-technical people
- One-click deployments (almost)
- Notifications; Slack messages look nice
- Scheduling Jobs (Playbook runs)
- Access Control

Cons:

- Free tier of 10 nodes, hard to determine value in
- Can not be triggered vi Webhooks
- Addresses many CI/CD needs, but still needs a CI
- Secrets get split between your repository and ansible tower,

Tooling: Prometheus Pros/Cons

Pros:

- Light, Quick setup
- Simple config file configuration
- Lots of integration,
- Very Open-Source , even the exporter is trying to standardize
- Large community

  CNCF member,

  Big : Digital Ocean, Soundcloud, Cloudflare, and now DB

Cons:

- Early Days, version 1.x to 2.x

  lots of changes, It can break!
- Dataloss, only recently a backup strategy
- Alertmanager and Prometheus UI not consolidated
- Long-term data is not it’s streatch
- No access control (but no a focus)

Addressing the challenges: *Making it dynamic*

Exploiting AWS Spot instance,
70% cheaper!! /w Block Statement

- Wrapping the ec2 module in a block statement and iterate over aws size type, Bid a set price: e.g **0.29~0.35** cent
- But with a condition: **“--extra-vars=aws_spotinstance=true”** as it can waste you more time

- ec2instances.info is great help for comparing

```
- block:
  - name: attemping to create a ec2 spot instance
    ec2:
      aws_access_key: "{{ aws_access_key }}"
      aws_secret_key: "{{ aws_secret_key }}"

      instance_type: "{{ item }}"
      image: "{{ ami.results[0].ami_id }}"
      region: "{{ aws_region }}"
      zone: eu-west-1c
      spot_price: 0.29
      spot_type: one-time
      spot_wait_timeout: 600
      wait: yes
      wait_timeout: 800
    register: ec2_deployment
    when: aws_spotinstance is defined and aws_spotinstance|bool == true
    with_items:
      - i3.4xlarge
      - i3.2xlarge
      - c3.xlarge
    register: ec2_deployment
```

Making it dynamic

Check in the playbook if your service is up, before adding it to monitoring and waking up people

```
TASK [dreamliner-run : print container results] ******************************
ok: [dreamliner] => {
    "msg": "Docker Container ID: b445b28b0c74 is using docker image 088918808992.dkr.ecr.eu-west-1.amazonaws.co
:v1.0.12-6 with IP: {u'macaddress': u'02:2b:21:2a:21:0d', u'network': u'10.0.2.0', u'mtu': 1500, u'broadcast':
as': u'enp0s3', u'netmask': u'255.255.255.0', u'address': u'10.0.2.15', u'interface': u'enp0s3', u'type': u'eth
0.0.2.2'}"
}

TASK [dreamliner-run : notify slack channel of progress] **********************
ok: [dreamliner] => {"changed": false, "msg": "OK"}

TASK [dreamliner-run : wait for app port to become avalaible] *****************
ok: [dreamliner] => {"changed": false, "elapsed": 6, "path": null, "port": 8080, "search_regex": null, "state":

TASK [dreamliner-run : wait and retry for healthcheck to responce HTTP 200] ****
FAILED - RETRYING: wait and retry for healthcheck to responce HTTP 200 (16 retries left).
FAILED - RETRYING: wait and retry for healthcheck to responce HTTP 200 (15 retries left).
FAILED - RETRYING: wait and retry for healthcheck to responce HTTP 200 (14 retries left).
FAILED - RETRYING: wait and retry for healthcheck to responce HTTP 200 (13 retries left).
```

Making it dynamic..

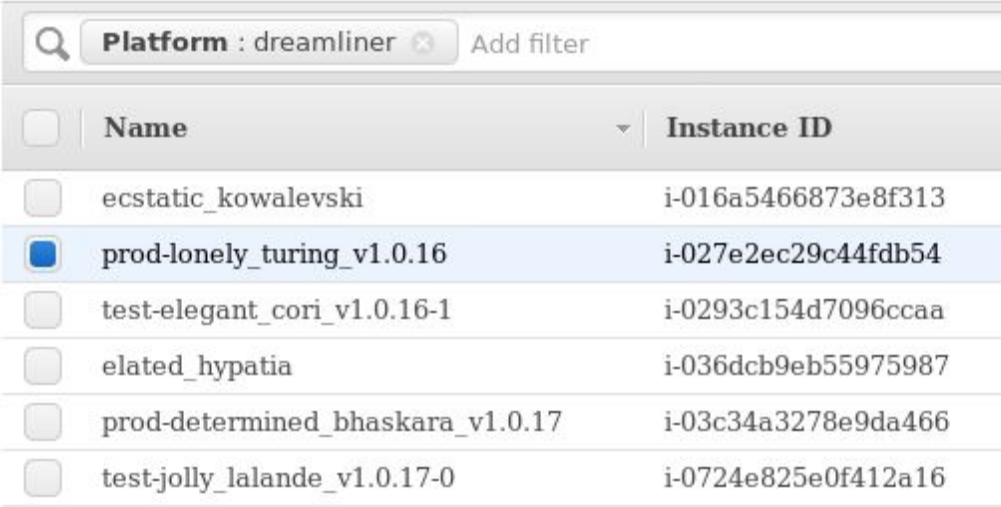
Use Tags, like everywhere, but not crazy

- Helps to maintain state
- Relabel your instances, use friendly names.
- Reference with environment and version

  “tag:Name=prod-elegant_cori_v1.0.16”
- Use instance filters in Ansible-Tower to consolidate your

  “tag:Platform=dreamliner”

AWS EC2 console: list of instances



Thank you

Questions

Cloud Guy, Trolls Movie, DreamWorks

My Details:
Darragh Grealish
Twitter: @grealish
56K.Cloud - DevOps Consulting and Services

For: Ansible, Docker, Network Infrastructure

dg@56k.cloud
https://www.56K.cloud/

No clouds were harmed during the deployment of this talk :)

Backup 2 - check application state before monitor

```
TASK [dreamliner-run : print container results] ******************************
ok: [dreamliner] => {
    "msg": "Docker Container ID: b445b28b0c74 is using docker image 088918808992.dkr.ecr.eu-west-1.amazonaws.com/dreamliner/sp
:v1.0.12-6 with IP: {u'macaddress': u'02:2b:21:2a:21:0d', u'network': u'10.0.2.0', u'mtu': 1500, u'broadcast': u'10.0.2.255',
as': u'enp0s3', u'netmask': u'255.255.255.0', u'address': u'10.0.2.15', u'interface': u'enp0s3', u'type': u'ether', u'gateway'
```

```
177 - name: wait and retry for healthcheck to responce HTTP 200
178   uri:
179     url: "http://{{ ansible_ec2_public_ipv4 }}:{{ dreamliner_app_port }}{{ dreamliner_app_healthcheck }}"
180     #body_format: json
181     return_contents: yes
182   register: healthcheck_responce
183   until: not healthcheck_responce|failed
184   retries: 16
185   delay: 4
186   ignore_errors: no
187   tags:
188     - deploy
189     - docker
190
```

Backup - looking up active targets in prometheus

```
12  post_tasks:
13    - name: wait for prometheus api target to come respond
14      uri:
15        uri: "http://{{ promethus_server_ipv4 }}:{{ promethus_port }}/{{ prom_api_query }}"
16        body_format: json
17        return_contents: yes
18      register: promethus_responce
19      until: not prometheus_responce|failed
20      retries: 16
21      delay: 4
22      ignore_errors: no
23      tags:
24        - deploy
25        - monitoring
26
27    - name: check if prometheus_responce contains status up for target
28      fail:
29        msg: "target is not up"
30      when: "'up' in prometheus_responce.json.data.activetargets.[iterate].health"
31      tags:
32        - monitoring
```

Backup - Before prometheus reload with targets

Ansible Tower

Not secure | https://192.168.106.15/#/home/hosts

TOWER

DASHBOARD / HOSTS

HOSTS 6

SEARCH | KEY

NAME	STATUS	ACTIONS
34.249.47.198	ON	
52.209.114.9	ON	
52.213.114.180	ON	
52.215.36.143	ON	
52.51.147.76	ON	
localhost	ON	

ITEMS 1 - 6 OF 6

Prometheus Time S

192.168.106.27:9090/targets

Prometheus Alerts Graph Status Help

Targets

node (2/2 up)

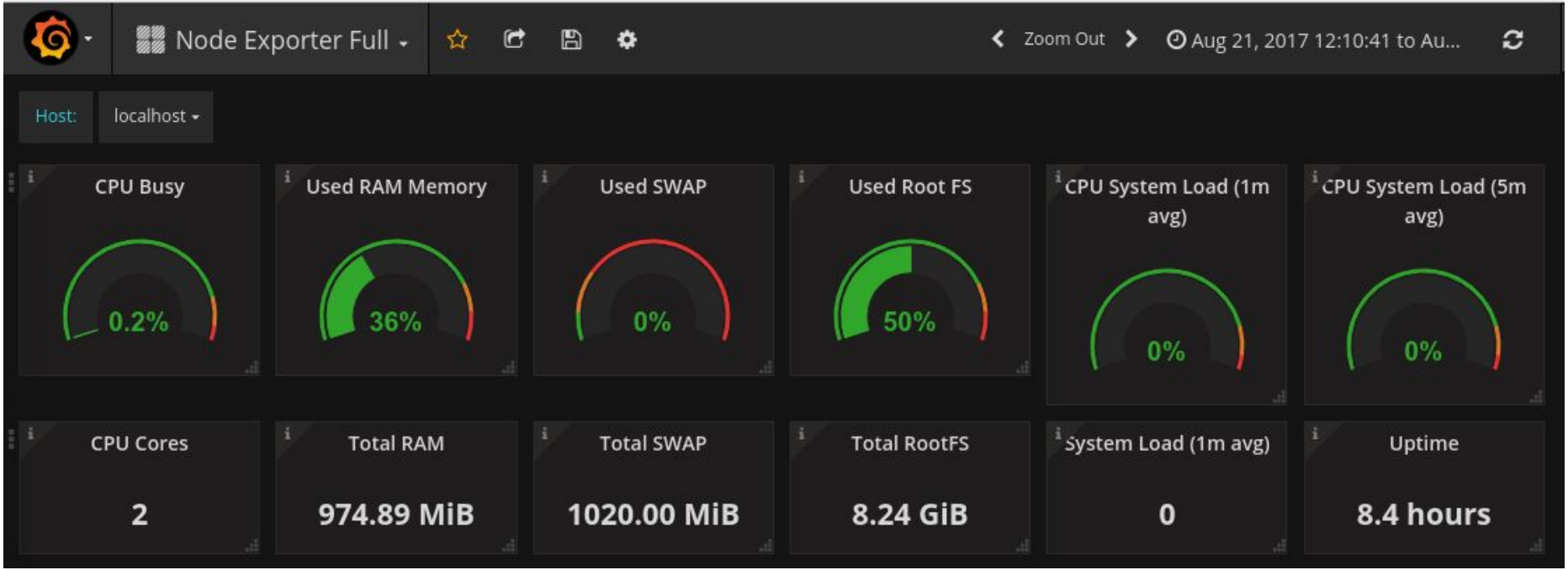
Endpoint	State	Labels	Last Scrape	Error
http://34.249.47.198:9100/metrics	UP	instance="34.249.47.198:9100"	2.26s ago	
http://localhost:9100/metrics	UP	instance="localhost:9100"	4.563s ago	

prometheus (1/1 up)

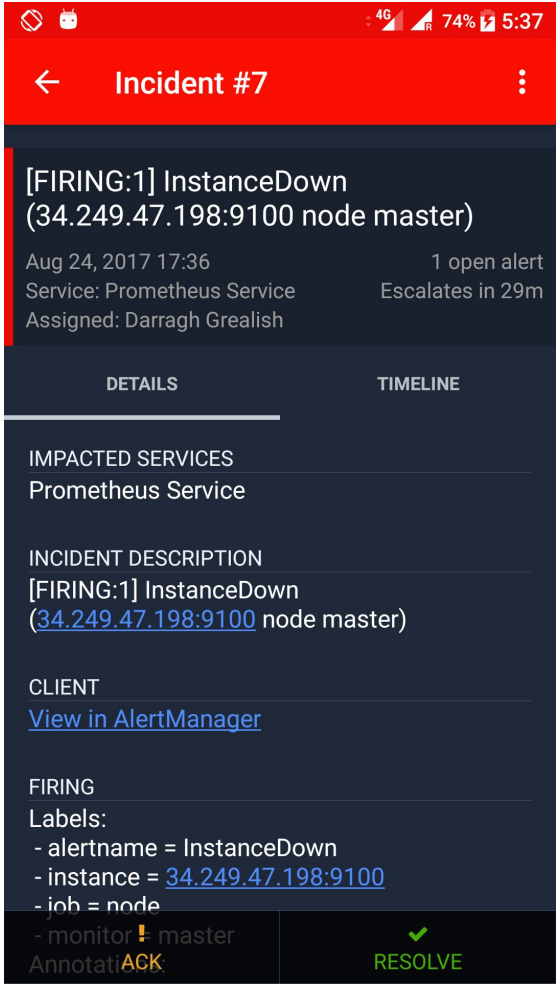
Endpoint	State	Labels	Last Scrape	Error
http://192.168.106.27:9090/metrics	UP	instance="192.168.106.27:9090"	6.001s ago	

Name	application	Instance Type	IPv4 Public IP
elated_hypatia	spectra-logging	t2.small	-
prod-lonely_turing_v1.0.16	spectra	t2.2xlarge	34.249.47.198
test-elegant_cori_v1.0.18-0	spectra	t2.medium	52.51.147.76
prod-determined_bhaskara_v1.0.17	spectra	t2.2xlarge	52.209.114.9
ecstatic_kowalevski	spectra	r3.4xlarge	52.213.114.180
test-jolly_lalande_v1.0.17-0	spectra	t2.medium	52.215.36.143

Backup - Grafana node exporter



Backup


74% 5:37
Incident #7
[FIRING:1] InstanceDown
(34.249.47.198:9100 node master)
Aug 24, 2017 17:36
1 open alert
Service: Prometheus Service
Escalates in 29m
Assigned: Darragh Grealish
DETAILS
TIMELINE
IMPACTED SERVICES
Prometheus Service
INCIDENT DESCRIPTION
[FIRING:1] InstanceDown
(34.249.47.198:9100 node master)
CLIENT
View in AlertManager
FIRING
Labels:
- alertname = InstanceDown
- instance = 34.249.47.198:9100
ACK
RESOLVE

Tower API Inventory

https://192.168.106.15/api/v1/inventories/67/hosts/

- We get a list of host
- Identify the hosts list
- Create a fact dict of hosts,
- Install prometheus exporters on these hosts
- Add the hosts into prometheus targets configuration
- Check for the targets to go green in the prometheus API

References:

Sources that supported this talk/demo:

https://prometheus.io/blog/2017/06/21/prometheus-20-alpha3-new-rule-format/