Richard Kiene @shmeeny
Tim Gross @0x74696d

Joyent

TRITON™

FULL STACK METRICS:
NATIVE PROMETHEUS SUPPORT ON TRITON

Joyent®

Modern Applications and Operations Made Easy

Node.js
Production Support

TRITON
Containers as a Service

Manta
Object Storage

Public Cloud

Triton Elastic Container Service. We run our customer's mission critical applications on container native infrastructure

Private Cloud

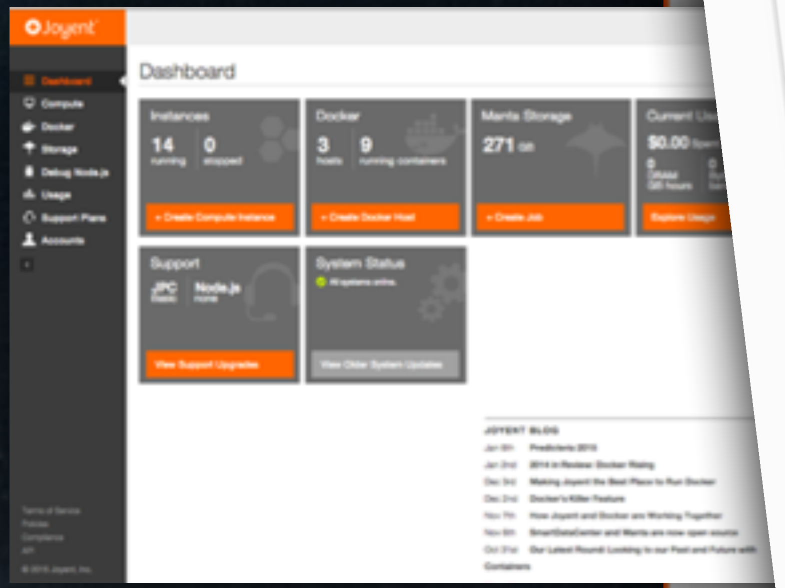
Triton Elastic Container Infrastructure is an on-premise, container run-time environment used by some of the world's most recognizable brands

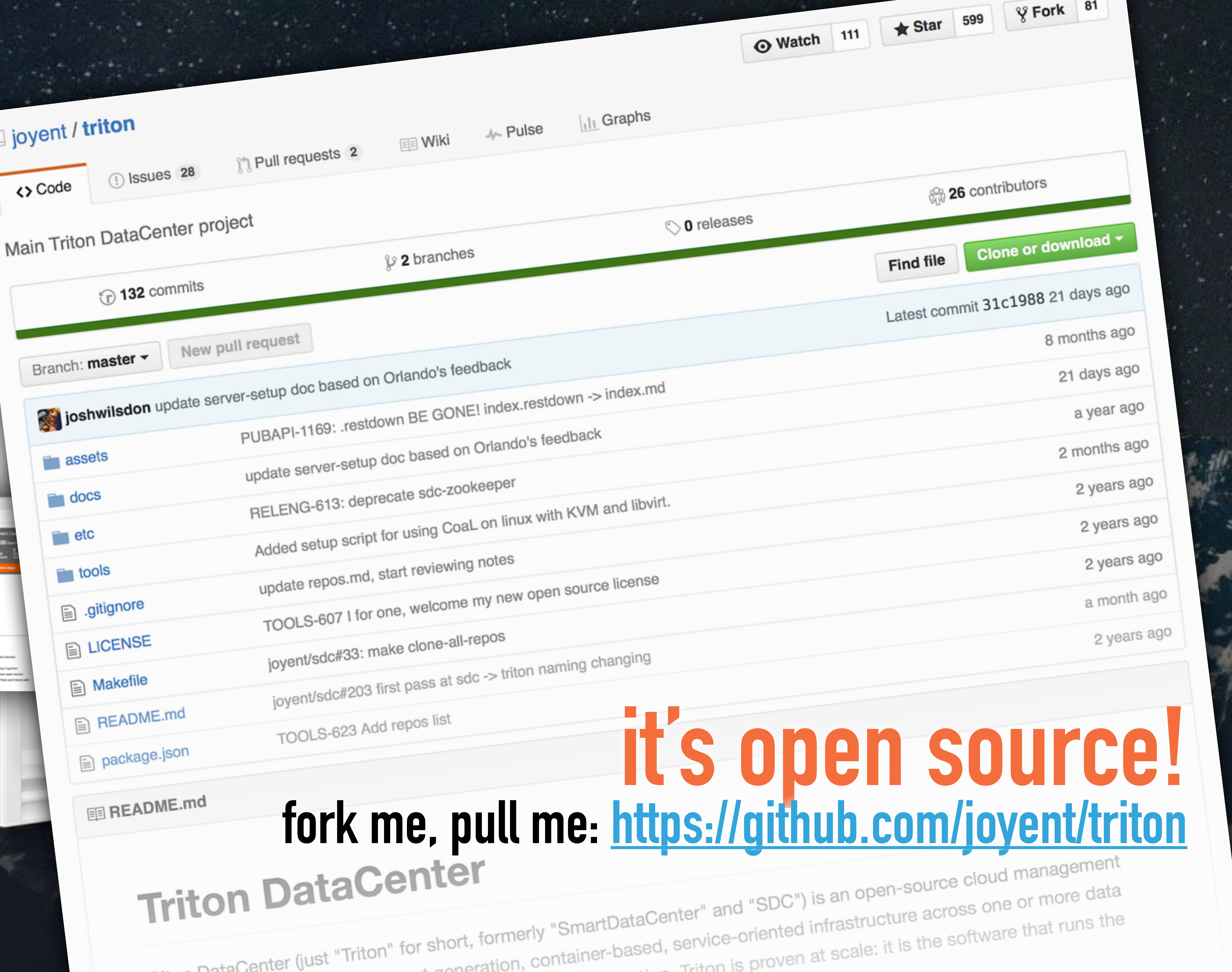
Public Cloud

Triton Elastic C

customer's missi

container native in





joyent / triton

Main Triton DataCenter project

132 commits | 2 branches | 0 releases | 26 contributors

joshwilsdon update server-setup doc based on Orlando's feedback — Latest commit 31c1988 21 days ago

assets	PUBAPI-1169: .restdown BE GONE! index.restdown -> index.md	8 months ago
docs	update server-setup doc based on Orlando's feedback	21 days ago
etc	RELENG-613: deprecate sdc-zookeeper	a year ago
tools	Added setup script for using CoaL on linux with KVM and libvirt.	2 months ago
.gitignore	update repos.md, start reviewing notes	2 years ago
LICENSE	TOOLS-607 I for one, welcome my new open source license	2 years ago
Makefile	joyent/sdc#33: make clone-all-repos	2 years ago
README.md	joyent/sdc#203 first pass at sdc -> triton naming changing	a month ago
package.json	TOOLS-623 Add repos list	2 years ago

README.md

Triton DataCenter

...DataCenter (just "Triton" for short, formerly "SmartDataCenter" and "SDC") is an open-source cloud management ...generation, container-based, service-oriented infrastructure across one or more data ... Triton is proven at scale: it is the software that runs the

it's open source!

fork me, pull me: https://github.com/joyent/triton

Come in...

WE'RE HIRING!

https://joyent.com/about/careers

> “We have built mind-bogglingly complicated systems that we cannot see, allowing glaring performance problems to hide in broad daylight in our systems.”

Bryan Cantrill, Joyent CTO

ACM Queue Vol 4, Issue 1, 2006 Feb 23

http://queue.acm.org/detail.cfm?id=1117401

> “System performance problems are typically introduced at the highest layers of abstraction, but they are often first encountered and attributed at the lowest layers of abstraction.”

Bryan Cantrill, Joyent CTO

ACM Queue Vol 4, Issue 1, 2006 Feb 23

http://queue.acm.org/detail.cfm?id=1117401

MONITORING IN PRODUCTION

- Hardest problems appear in production
- Must be able to observe safely in production:
  - No risk of crashing
  - Dynamic instrumentation: no performance hit on observed environment

VALUE OF OBSERVABILITY

- Observability is the key to being production-ready
- Much of Joyent’s value over our competitors is our best-in-class observability and debugging tooling

TRITON ARCHITECTURE

- Customer applications run as containers
- SmartOS or Linux (LX) infrastructure containers, or Docker application containers, running as Solaris Zones
- Proven battle-tested multi-tenant security
- Bare-metal performance
- Isolation provides observability w/o interference

CLOUD ANALYTICS V1

- Historical data is cumbersome to use
- API is awkward for high-dimensionality
- Want to improve scalability w/ aggregation
- Want better availability
- No path for end users to application-level metrics

DESIGN CONSTRAINTS

- Multi-tenant:
  - Operators of Triton provide an API for customers (end-users, developers, etc.) to deploy their containers.
  - One customer can’t cause brown-outs for other customers!
- Give customers a sane migration path or let them use their existing monitoring

WHY PULL?

- We don’t drop metrics for overloaded target (collection happens outside the zone)
- Can easily throttle customer requests
- Pushing to a customer collector that’s down requires implementing back-off/buffering for every customer in metrics agent
- End-users can have multiple consumers

WHY PROMETHEUS?

- Pull not push
- Agnostic to storage: end-users can do what they want with the metrics afterwards (including push them into their existing metrics solution if they want!)

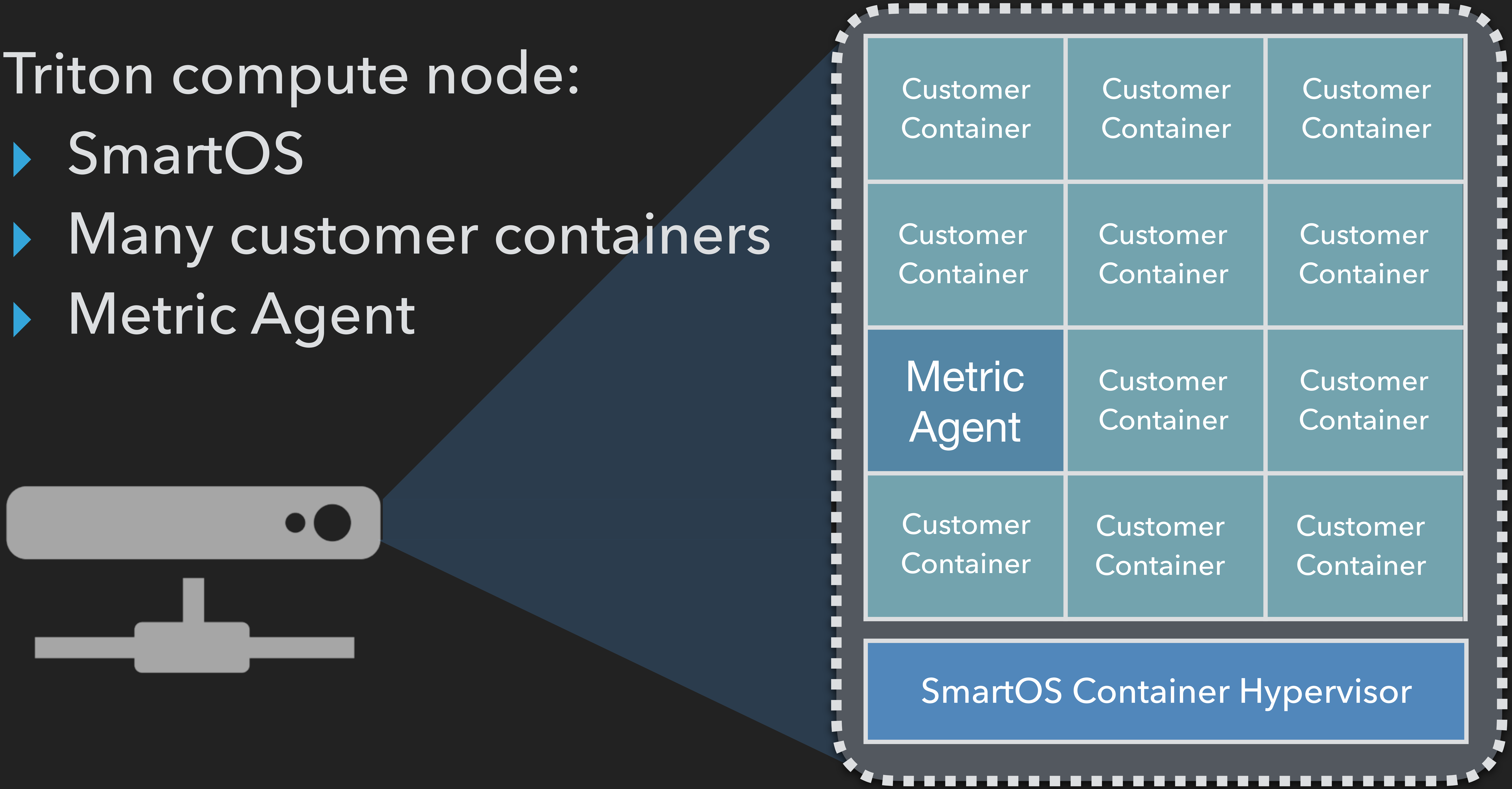
CONTAINER MONITOR: ARCHITECTURE

METRIC AGENT

- Instance on each physical machine (“compute node”)
- Collects metrics from all containers via kstat, zfs list, etc.

Triton compute node:

- SmartOS
- Many customer containers
- Metric Agent



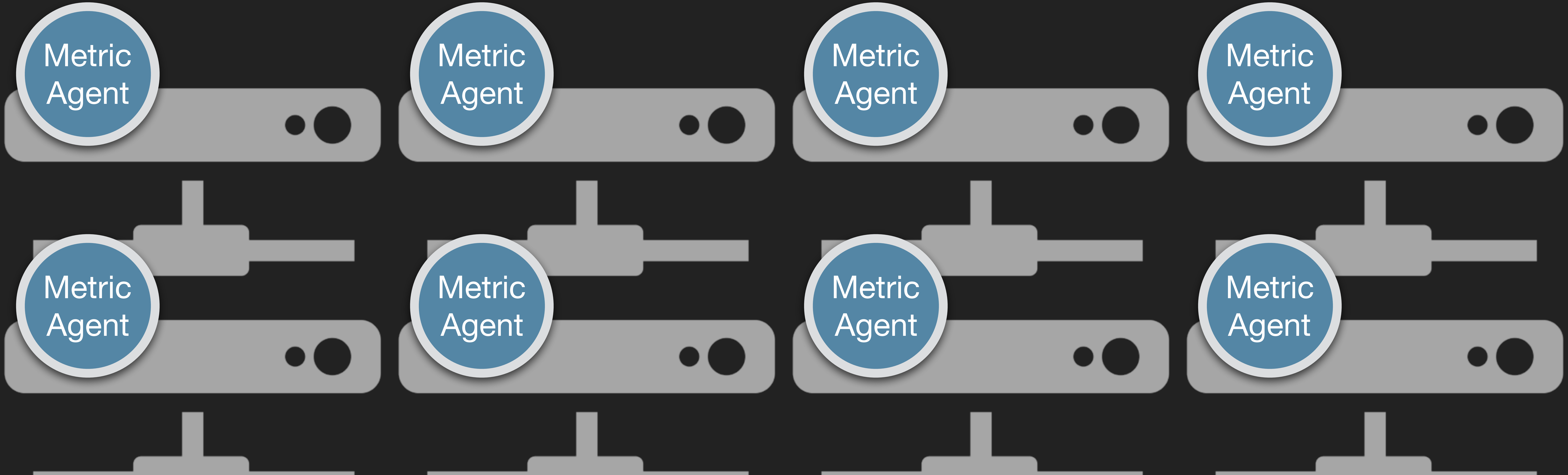
Customer Container	Customer Container	Customer Container
Customer Container	Customer Container	Customer Container
Metric Agent	Customer Container	Customer Container
Customer Container	Customer Container	Customer Container

SmartOS Container Hypervisor

Noun Project icon by Aneeque Ahmed

Triton data center:

- Many compute nodes
- Each has its own Metric Agent



METRIC AGENT PROXY

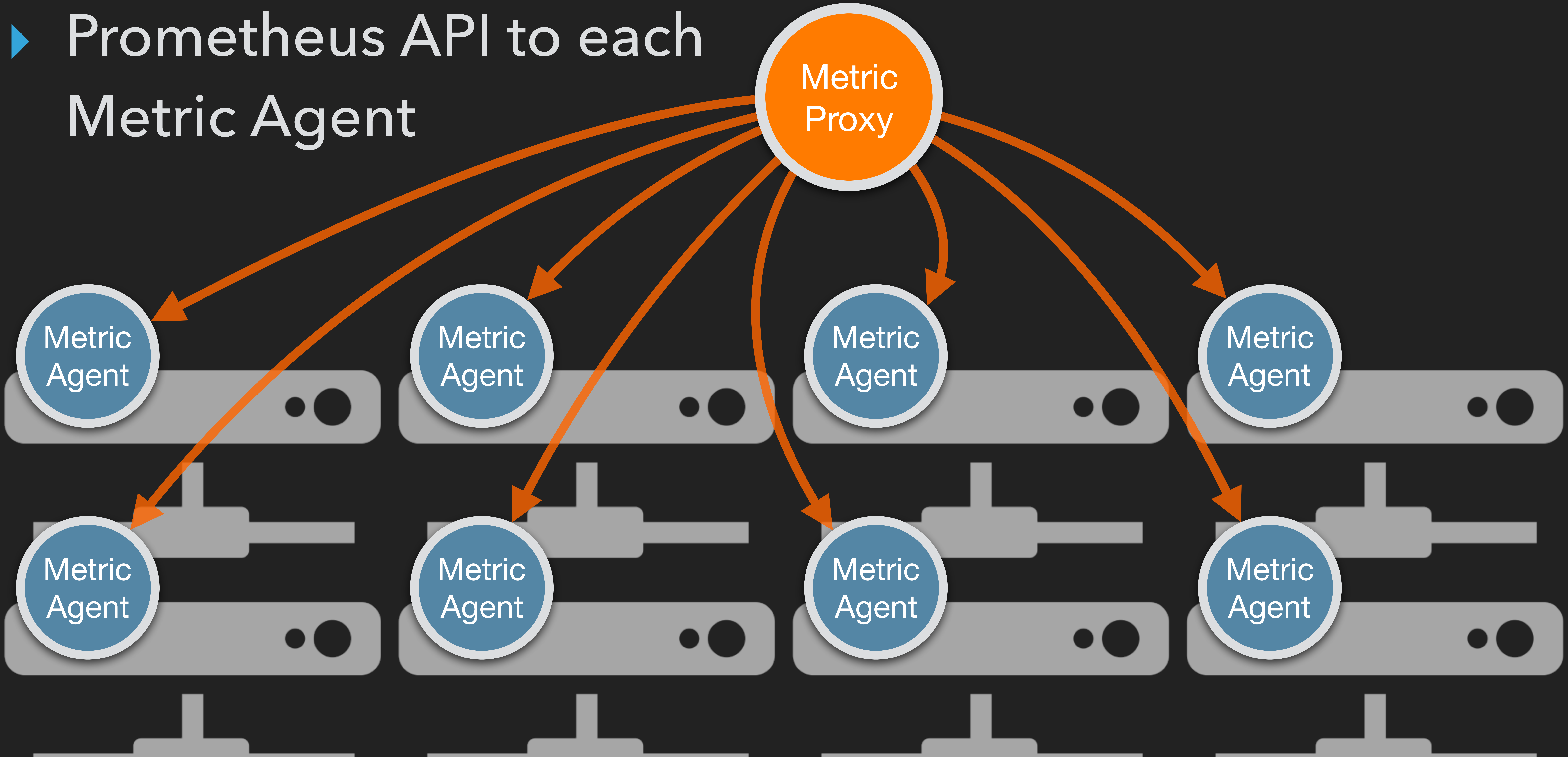
- Stateless and horizontally scalable
- HA across data center: 1 on head node + min 2 per DC
- Routes Prometheus server requests to appropriate Metric Agent
- Responsible for rate-limiting and authentication

DISCOVERY: TRITON CNS

- Triton Container Name Service (CNS): automated container-native DNS service
- Containers are automatically assigned A-Records for instances (and services)
- Container Monitor provides CNAME to Metric Agent Proxy’s IP for each container

Metric Agent Proxy:

- Prometheus API to each Metric Agent

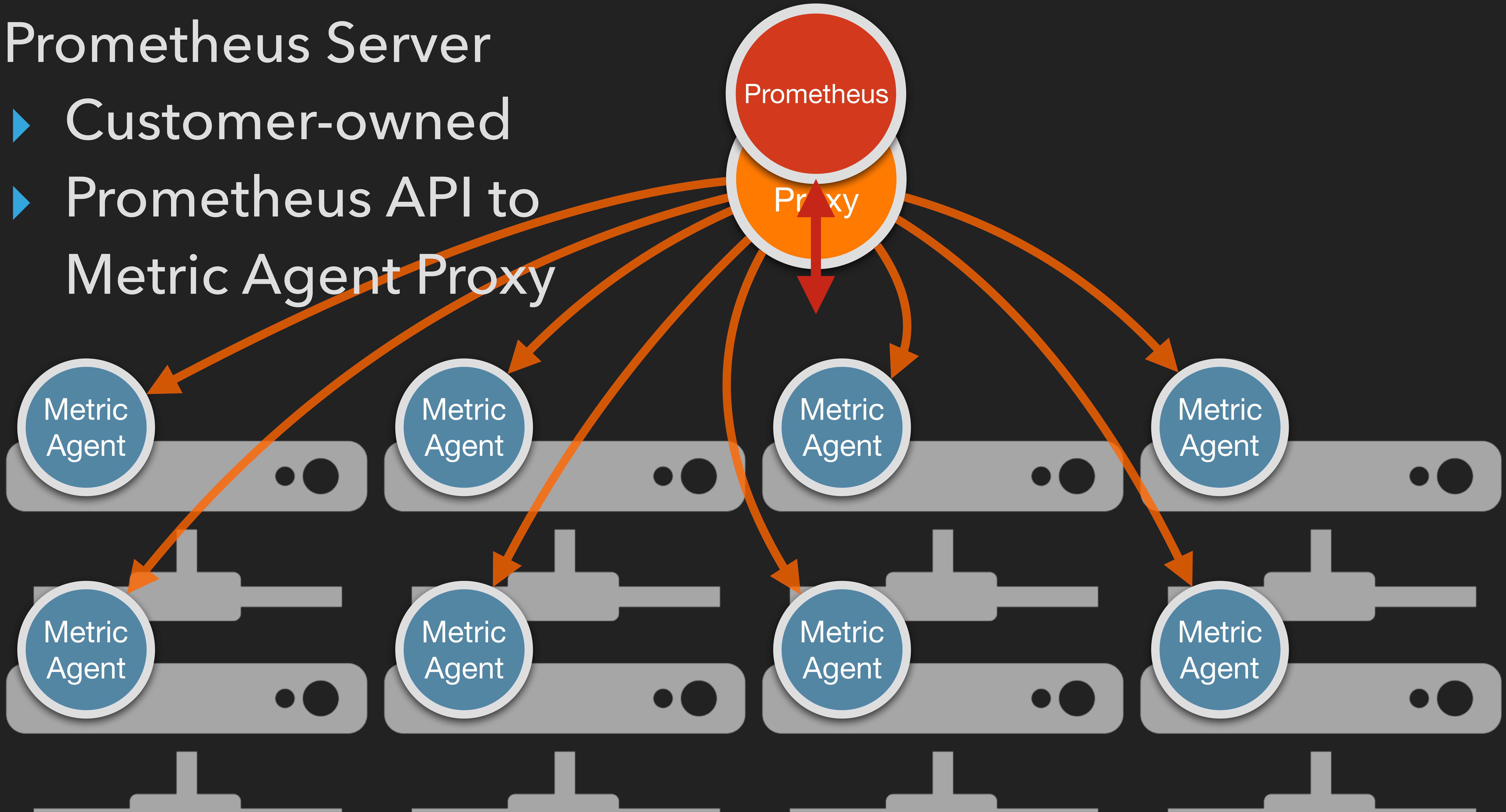


METRICS COLLECTION

- Customer-owned Prometheus server(s)
- Optional customer-owned Metrics Forwarders: forward metrics to existing monitoring systems

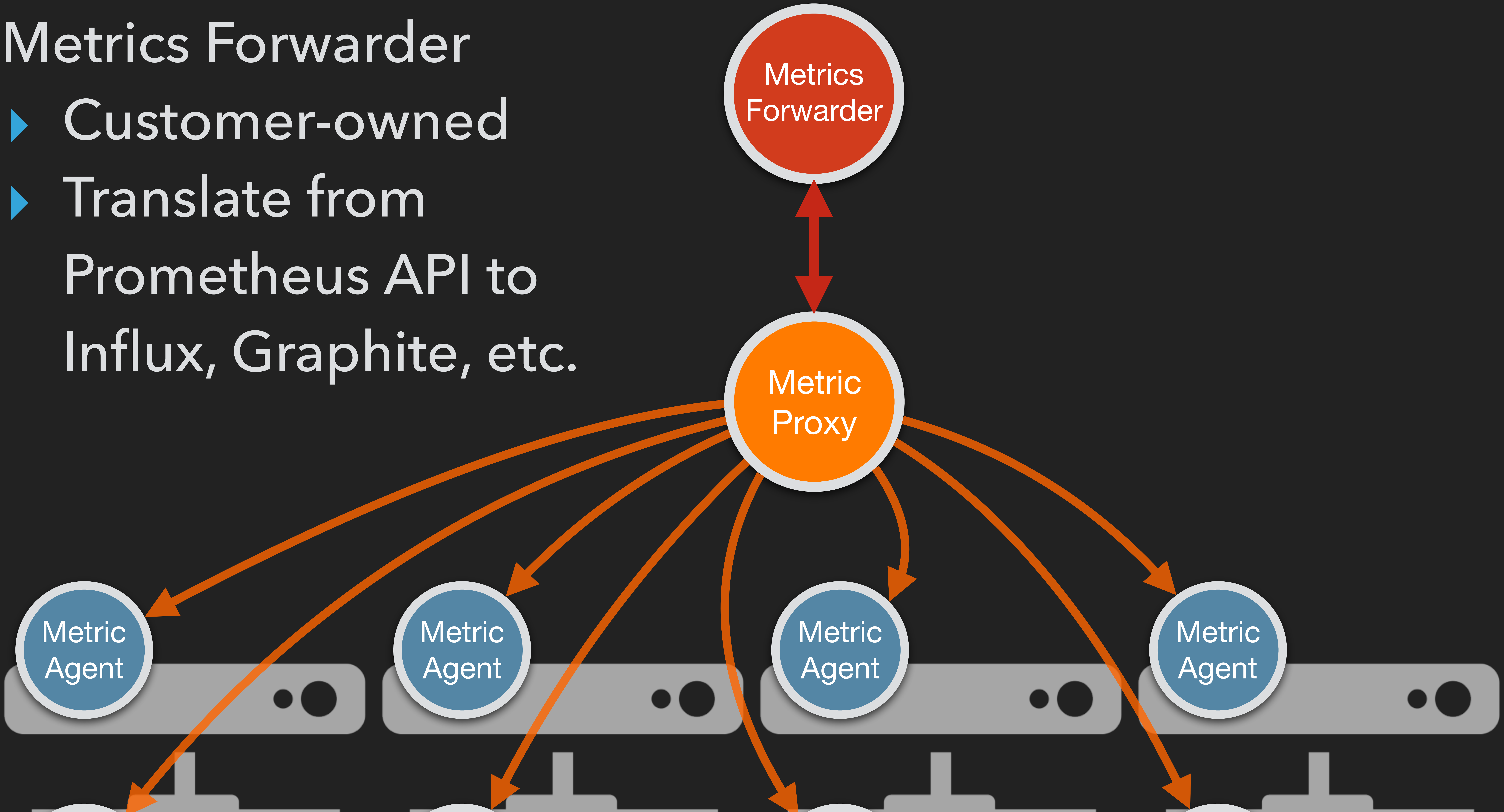
Prometheus Server

- Customer-owned
- Prometheus API to Metric Agent Proxy



Metrics Forwarder

- Customer-owned
- Translate from Prometheus API to Influx, Graphite, etc.



HOW A CONTAINER GETS MONITORED

- End-user launches container
- VMAPI pushes change feed event to CNS
- New CNAME record for each container to Metric Agent Proxy IP address

HOW A CONTAINER GETS MONITORED, CONT.

- Customer's Prometheus server uses Triton discovery plugin to poll metric agent proxy endpoints for all containers associated with that account
- Metric Agent Proxy forwards requests to appropriate metric agent

APPLICATION METRICS: CONTAINERPILOT

AUTOPILOT PATTERN

- Design pattern for self-operating and self-managing applications
- Containers adapt to changes in their environment and coordinate their actions thru globally shared state
- Platform agnostic

CONTAINERPILOT

- App-centric micro-orchestrator that enables the Autopilot Pattern
- Acts as PID1 in the container and fires user-defined life-cycle hooks
- Telemetry “sensor” hooks feed data to a Prometheus metrics endpoint

CONTAINERPILOT METRICS ON TRITON

- Containers have a CNS name
- ContainerPilot exposes Prometheus endpoint
- Add discovery catalog (ex. Consul, etcd) to Prometheus server config

ContainerPilot config file

```
{
  "consul": "consul:8500",
  "preStart": "/usr/local/bin/reload.sh preStart",
  "logging": {"level": "DEBUG"},
  "services": [
    {
      "name": "nginx",
      "port": 80,
      "health": "/usr/bin/curl --fail -s http://localhost/health",
      "poll": 10,
      "ttl": 25
    }
  ],
  "backends": [
    {
      "name": "example",
      "poll": 7,
      "onChange": "/usr/local/bin/reload.sh"
    }
  ],
 "telemetry": {
    "port": 9090,
    "sensors": [
      {
        "name": "tb_nginx_connections_unhandled_total",
        "help": "Number of accepted connnections that were not handled",
        "type": "gauge",
        "poll": 5,
        "check": ["/usr/local/bin/sensor.sh", "unhandled"]
      },
      {
        "name": "tb_nginx_connections_load",
        "help": "Ratio of active connections (less waiting) to the maximum worker connections",
        "type": "gauge",
        "poll": 5,
        "check": ["/usr/local/bin/sensor.sh", "connections_load"]
      }
    ]
  }
}
```

ContainerPilot config file

```
{
  "consul": "consul:8500",
  "preStart": "/usr/local/bin/reload.sh preStart",
  "logging": {"level": "DEBUG"},
  "services": [
    {
      "name": "nginx",
      "port": 80,
      "health": "/usr/bin/curl --fail -s http://localhost/health",
      "poll": 10,
      "ttl": 25
    }
  ],
  "backends": [
    {
      "name": "example",
      "poll": 7,
      "onChange": "/usr/local/bin/reload.sh"
    }
  ],
 “telemetry”: {
     "port": 9090,
     "sensors": [
       {
         "name": "tb_nginx_connections_unhandled_total",
         "help": "Number of accepted connnections that were not handled",
```

ContainerPilot config file

```
  "backends": [
    {
      "name": "example",
      "poll": 7,
      "onChange": "/usr/local/bin/reload.sh"
    }
  ],
“telemetry”: {
    "port": 9090,
    "sensors": [
      {
        "name": "tb_nginx_connections_unhandled_total",
        "help": "Number of accepted connnections that were not handled",
        "type": "gauge",
        "poll": 5,
        "check": ["/usr/local/bin/sensor.sh", "unhandled"]
      },
      {
        "name": "tb_nginx_connections_load",
        "help": "Ratio of active connections (less waiting) to the maximum worker connections",
        "type": "gauge",
        "poll": 5,
        "check": ["/usr/local/bin/sensor.sh", "connections_load"]
      }
    ]
  }
}
```

DEMO

Richard Kiene @shmeeny

Tim Gross @0x74696d

Joyent

TRITON™

FULL STACK METRICS:
NATIVE PROMETHEUS SUPPORT ON TRITON