Dynatrace for Cloud Application Teams running microservices on k8s

dynatrace Perform

Steve Caron

Brandon Neo

Ken Silber

Alois Mayr

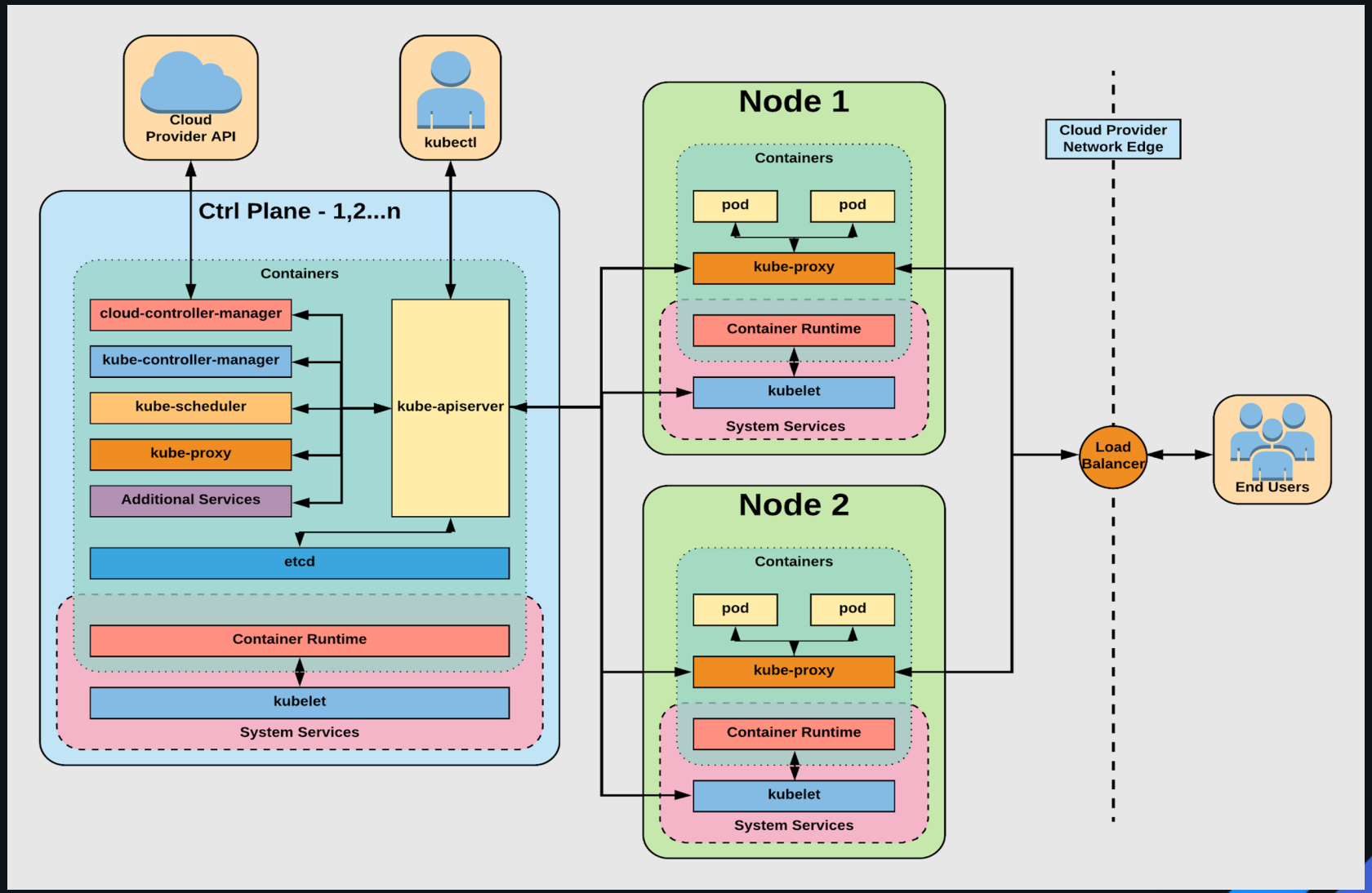
Agenda

1. Intro
   - K8s refresh
   - Dynatrace and k8s
   - Lab environment
2. Import k8s Labels & Annotations
3. Customize Service Naming
4. Play with Management Zones
5. Set up Alerting Profiles
6. Detect Performance Problems
7. Deploy a Canary
8. Import Prometheus Metrics
9. Manage your Workload Resource Usage

Show of hands : https://pollev.com/stevecaron983

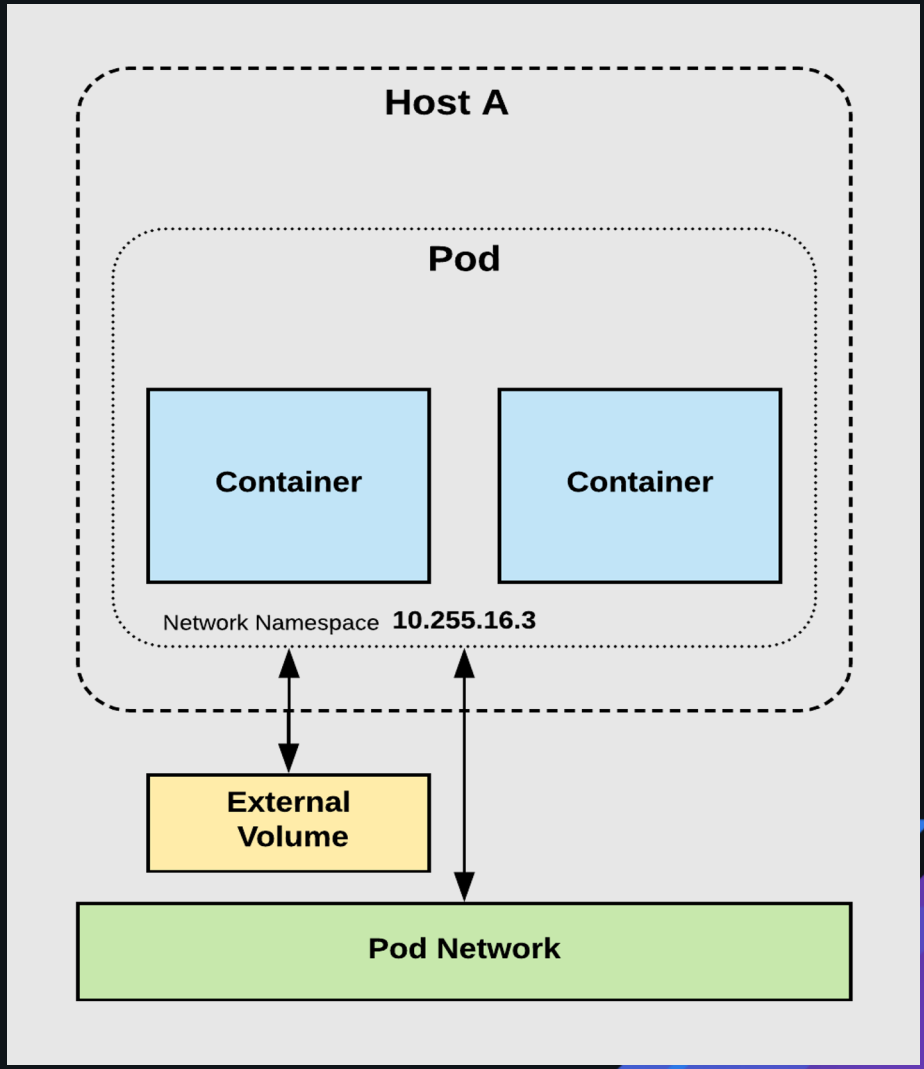
Kubernetes Concepts and Architecture

Architecture overview



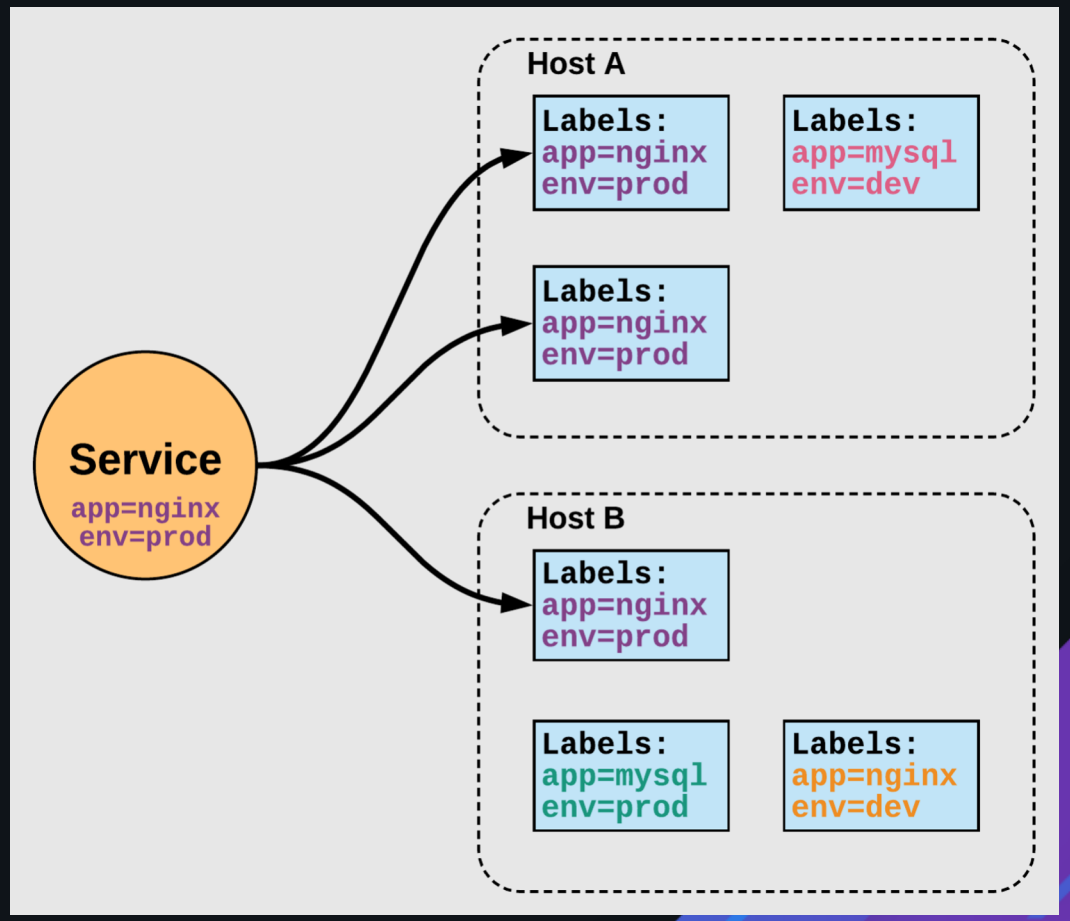
Key Concepts - Pods

- Smallest “unit of work” in Kubernetes.
- Pods are one or more containers sharing volumes and namespace.
- They are also ephemeral



Key Concepts - Services

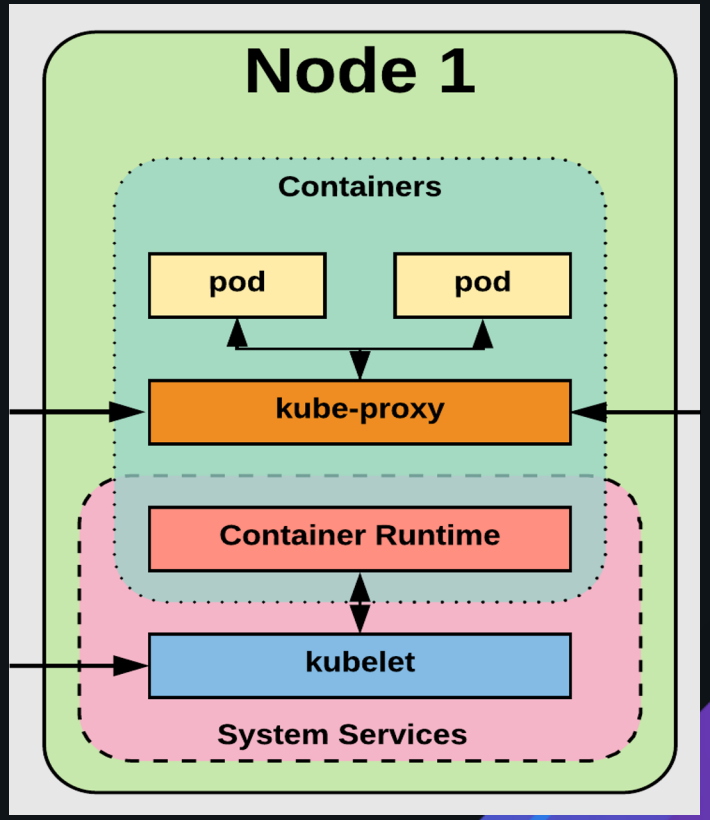
- Unified method of accessing the exposed workloads of pods.
- Think of it as an internal load balancer to your pods.
- Implementation depends cloud provider or on-prem config
- Not ephemeral



<service name> <namespace>.svc.cluster.local

Key Concepts - Nodes

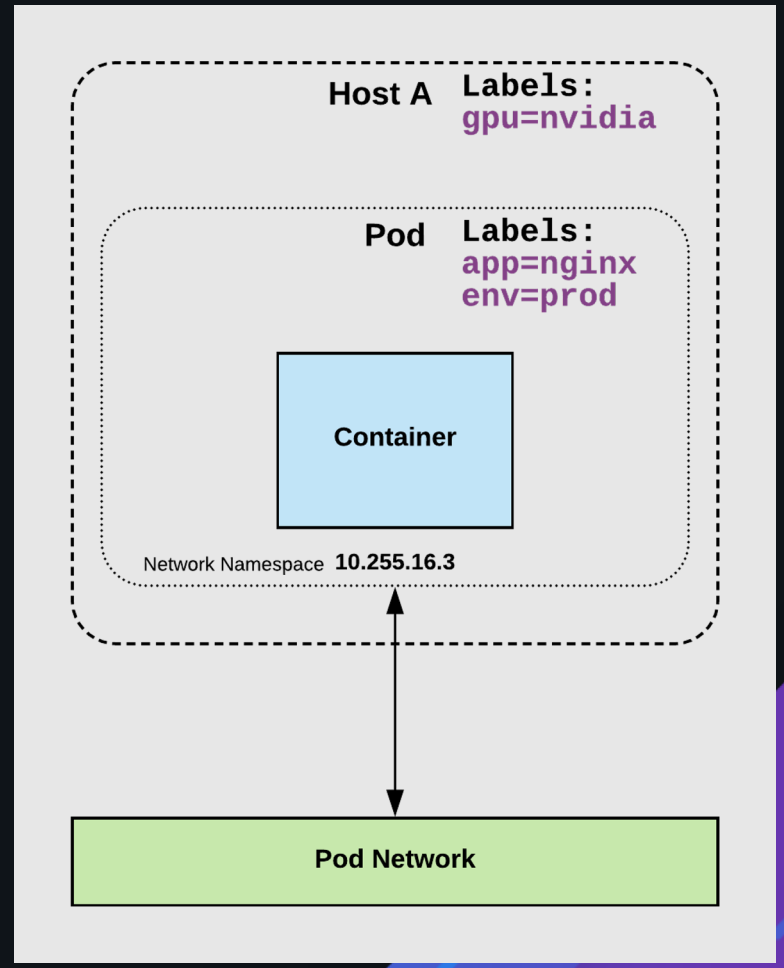
- Worker machine in Kubernetes
- VM or physical machine
- Managed by the control plane (master)
- Node components:
  - kubelet
  - kube-proxy
  - Container Runtime Engine



Key Concepts – Labels

- Key-value pairs used to identify, describe and group together related sets of objects or resources.

- Not characteristic of uniqueness

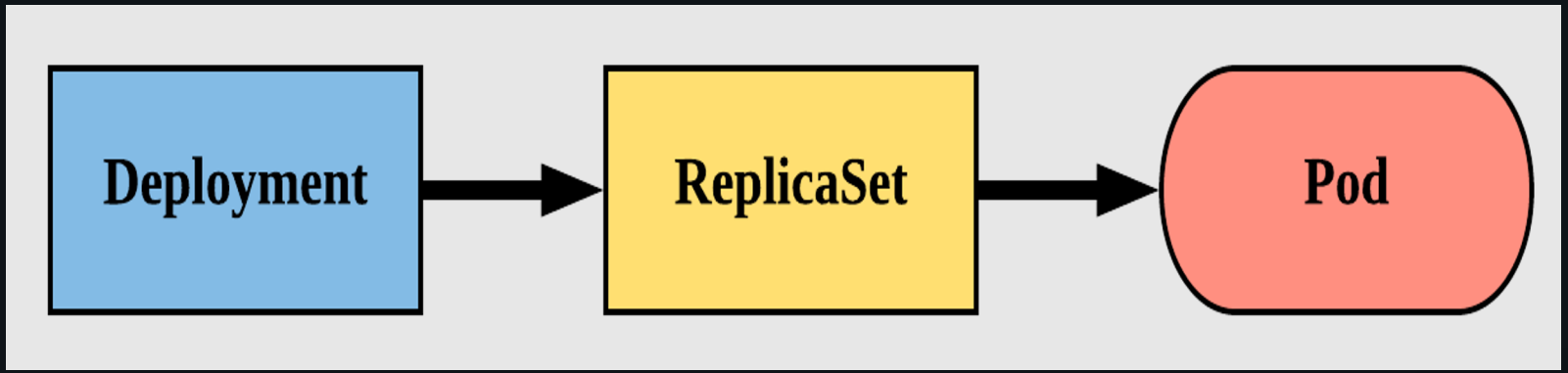


Key Concepts – Replicasets

- Primary method of managing pod replicas and their lifecycle
- Includes their scheduling, scaling, and deletion
- Their job is simple: Always ensure the desired number of pods are running

Key Concepts – Deployments

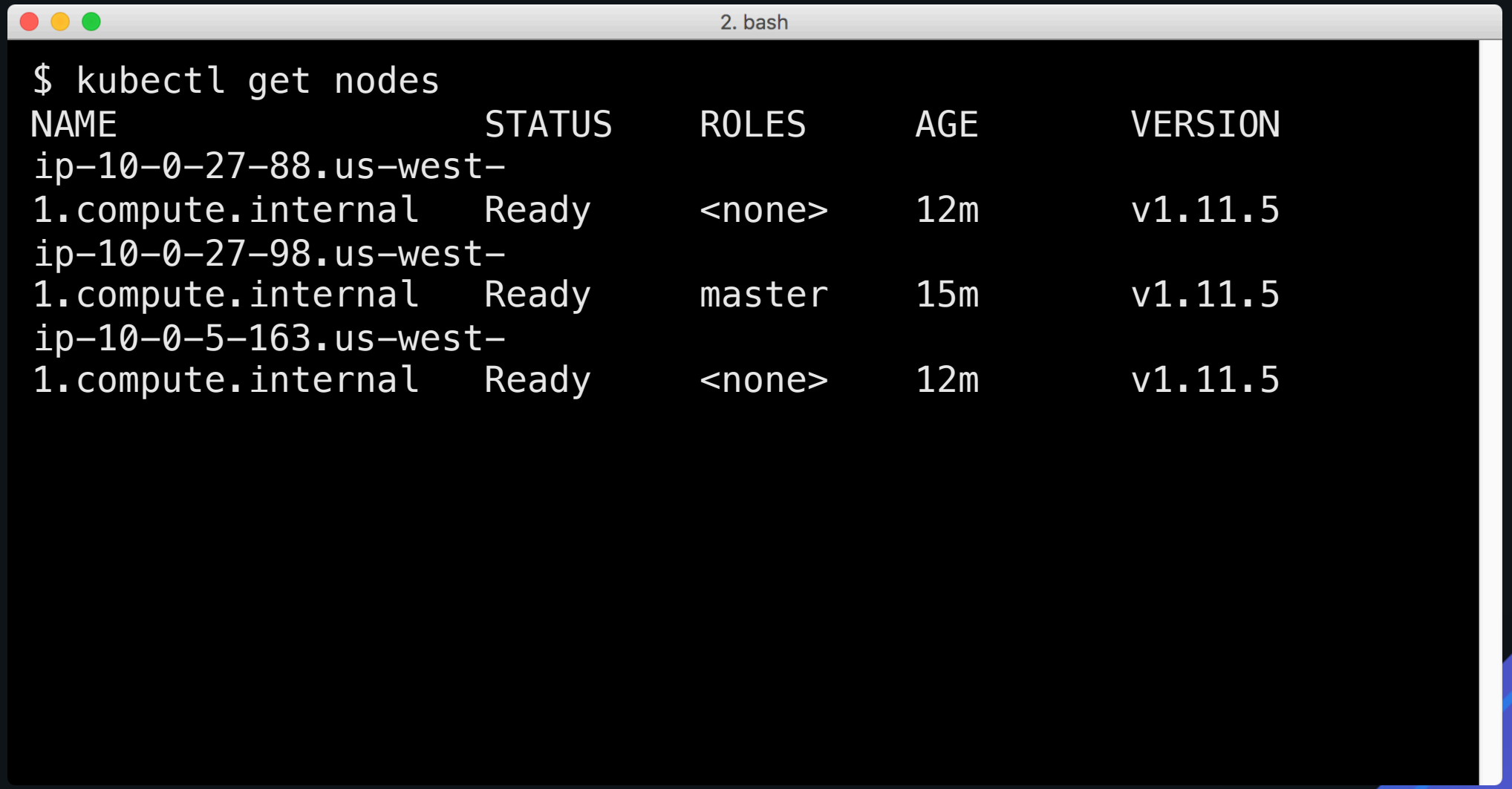
- Way of managing Pods via ReplicaSets
- Provide rollback functionality and update control
- Each iteration creates a unique label that is assigned to both the ReplicaSet and subsequent Pods.



Key Concepts – Daemonsets

- Ensure that all nodes matching certain criteria will run an instance of the supplied Pod
- Are ideal for cluster wide services such as log forwarding or monitoring

Key Concepts – kubectl (CLI)



Kubernetes Concepts Cheat sheet

Nodes
- are instances of underlying OS (typically Linux)
- virtual or physical machine (bare-metal k8s)
- host one to many *Pods*

Pods
- are (co-located) groups of *containers*
- a Pod's containers run *on the same host (node)*

Labels
- assign *identifying metadata* to *objects* (Pods)
- are used to organize, group or select objects

Annotations
- assign *non-identifying metadata* to objects
- are used for arbitrary data (build, debug info)

Deployments/Replicasets
- select a Pod object using *labels*
- ensure a given number of *Pod replicas*
- handle *scale-up/down* and *rolling updates*

Services
- select a Pod object using *labels*
- abstract a (replicated) set of Pods by *IP, port(s)*
- provide *domain name* and simple *load balancing*

Ingress
- expose services to the (cluster) outside world

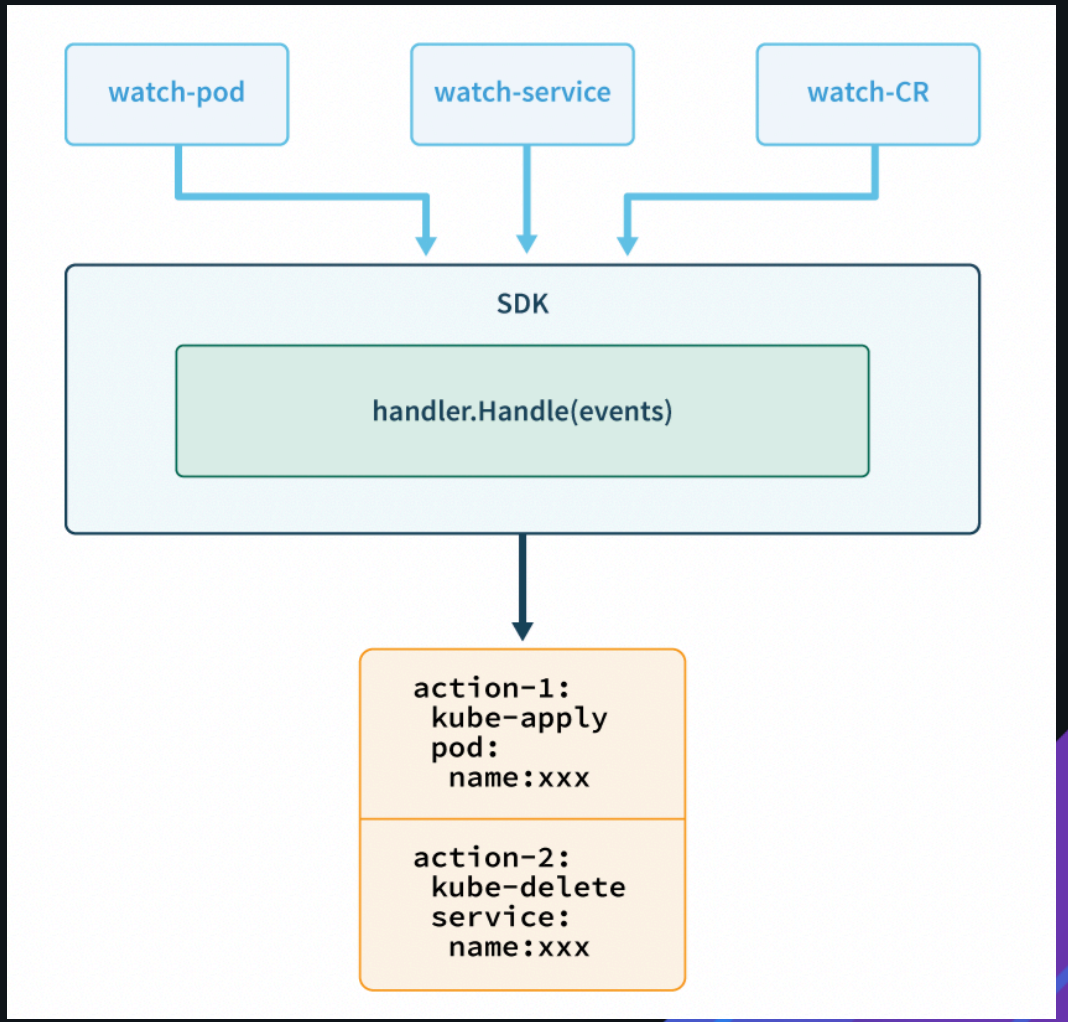
Kubernetes concepts also available from the CLI:
$ kubectl api-resources

Monitoring Kubernetes with the OneAgent Operator

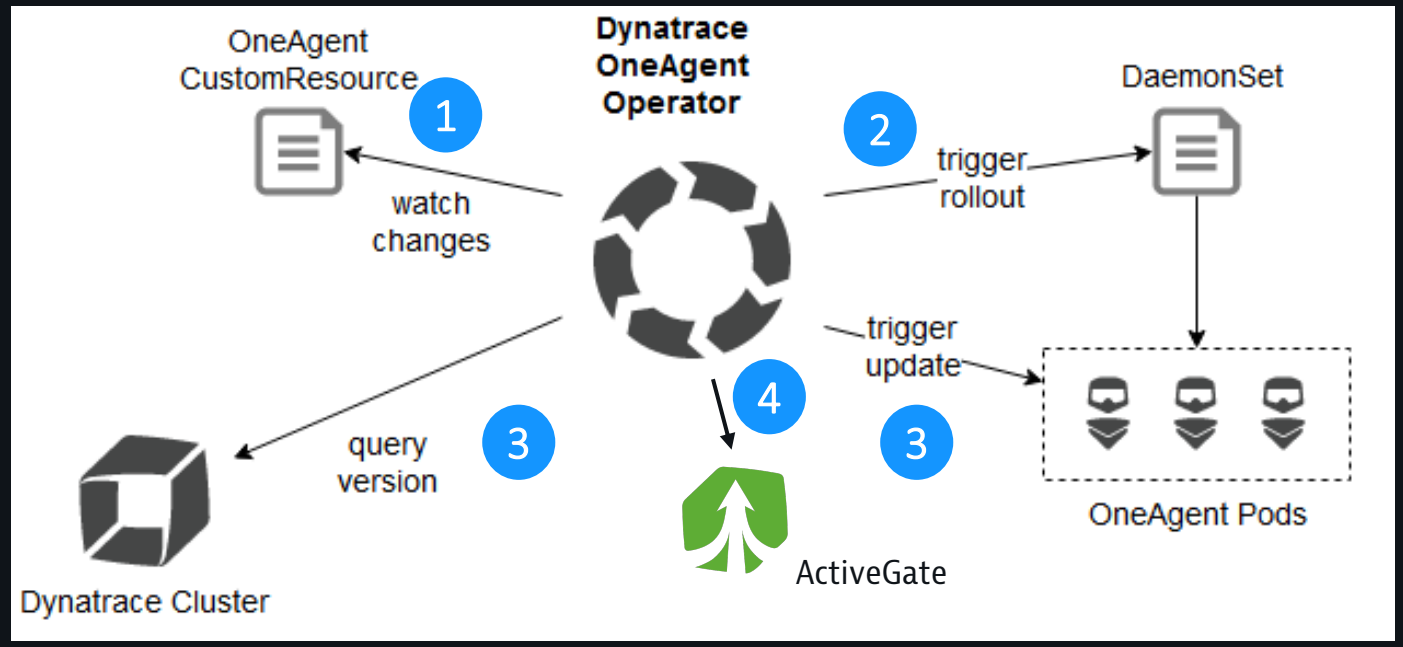
What is an Operator?

- Open source toolkit to manage Kubernetes native applications in an effective, automated, and scalable way.
- *"Human operational knowledge in software"*
- Reliably manage application lifecycles as a first-class kube native object

OPERATOR FRAMEWORK



OneAgent Operator in action

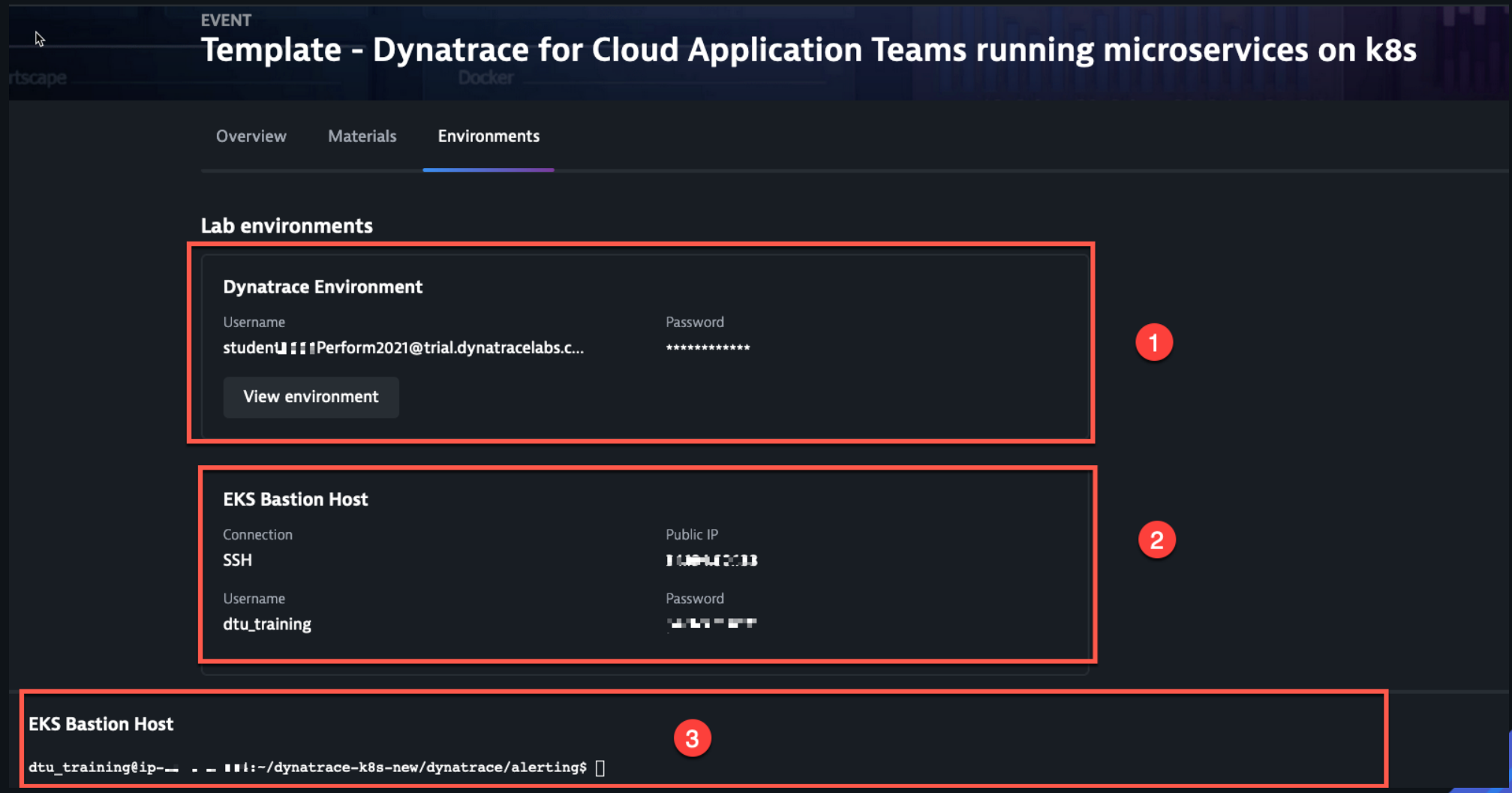


1. Watches for custom resources of type OneAgent
2. Takes care of OneAgent deployment via DaemonSet
3. Updates OneAgent to the latest version available
4. Deploys ActiveGate in k8s as StatefulSet (1.209)

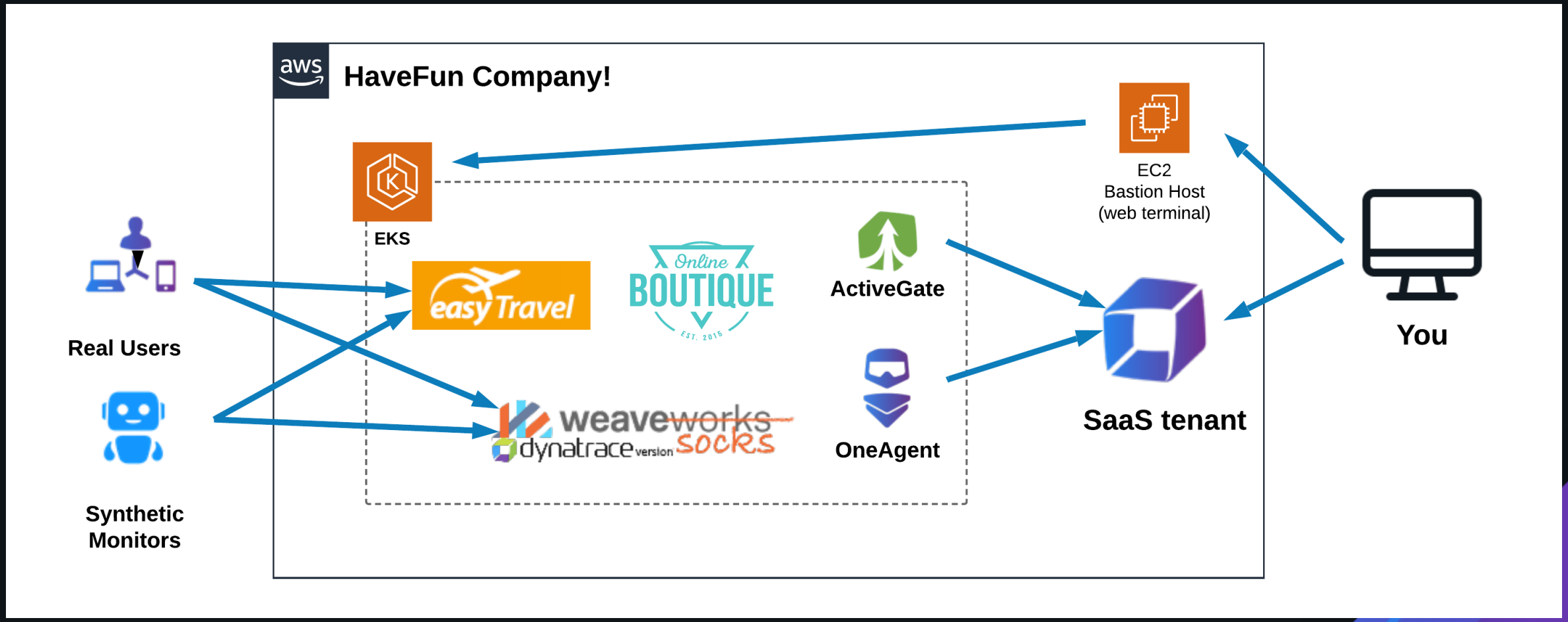
- Validated : AWS, Azure, GCP, Red Hat (and more..) certified!
- Can be deployed via Helm chart, OpenShift console or kubectl/ocp CLI
- https://github.com/Dynatrace/dynatrace-oneagent-operator

Excercise #1 Lab environment

Perform Environment

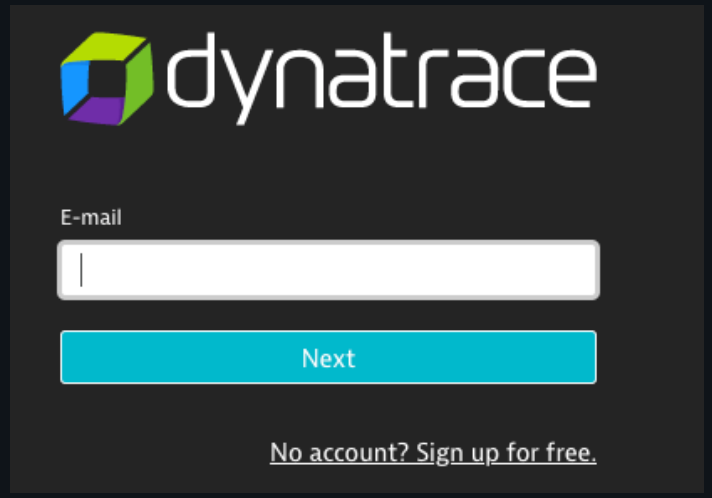


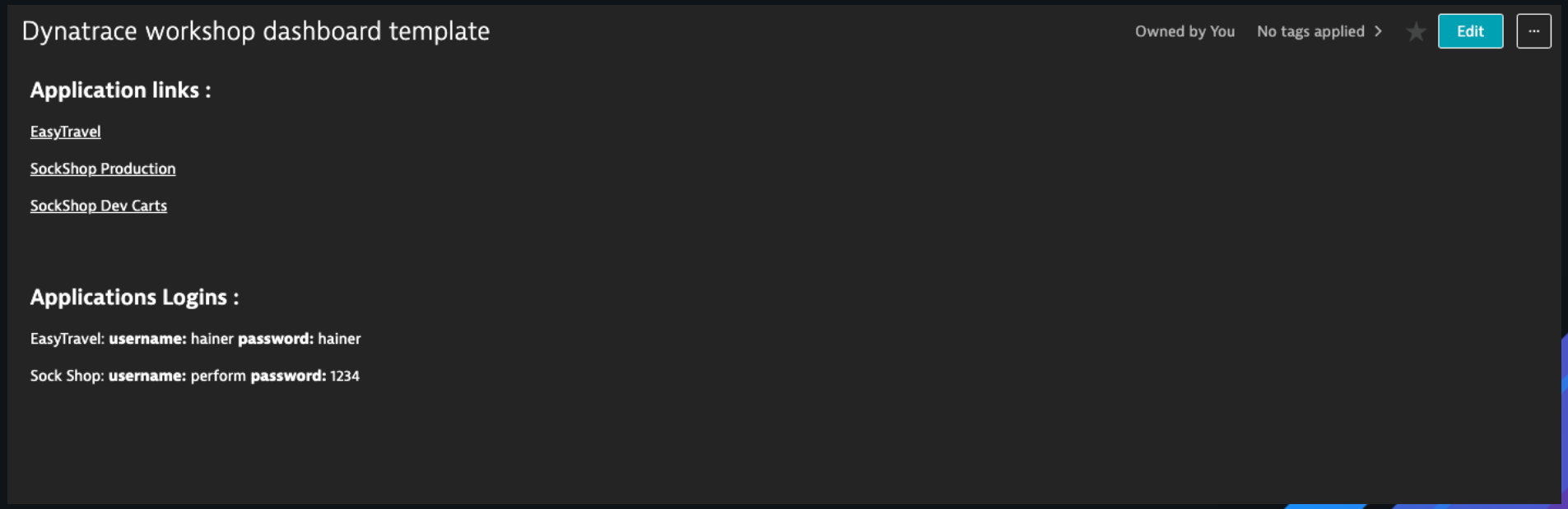
HaveFun Company setup!!!



Dynatrace

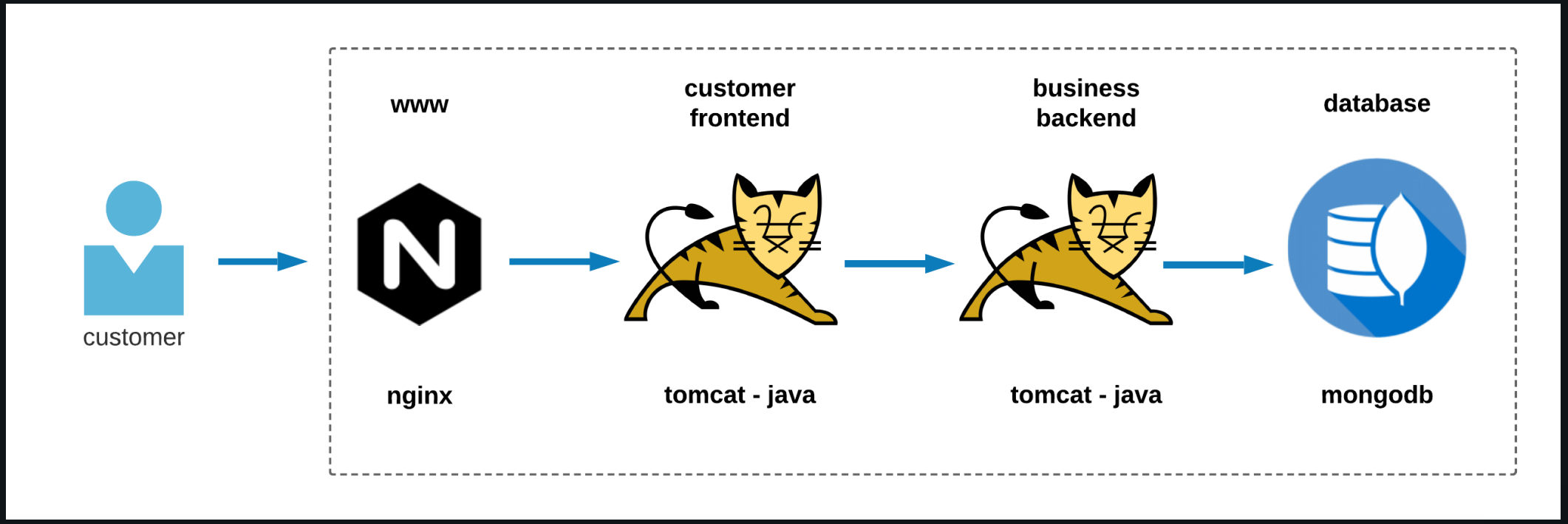
Your Dynatrace tenant





Easytravel

EasyTravel architecture

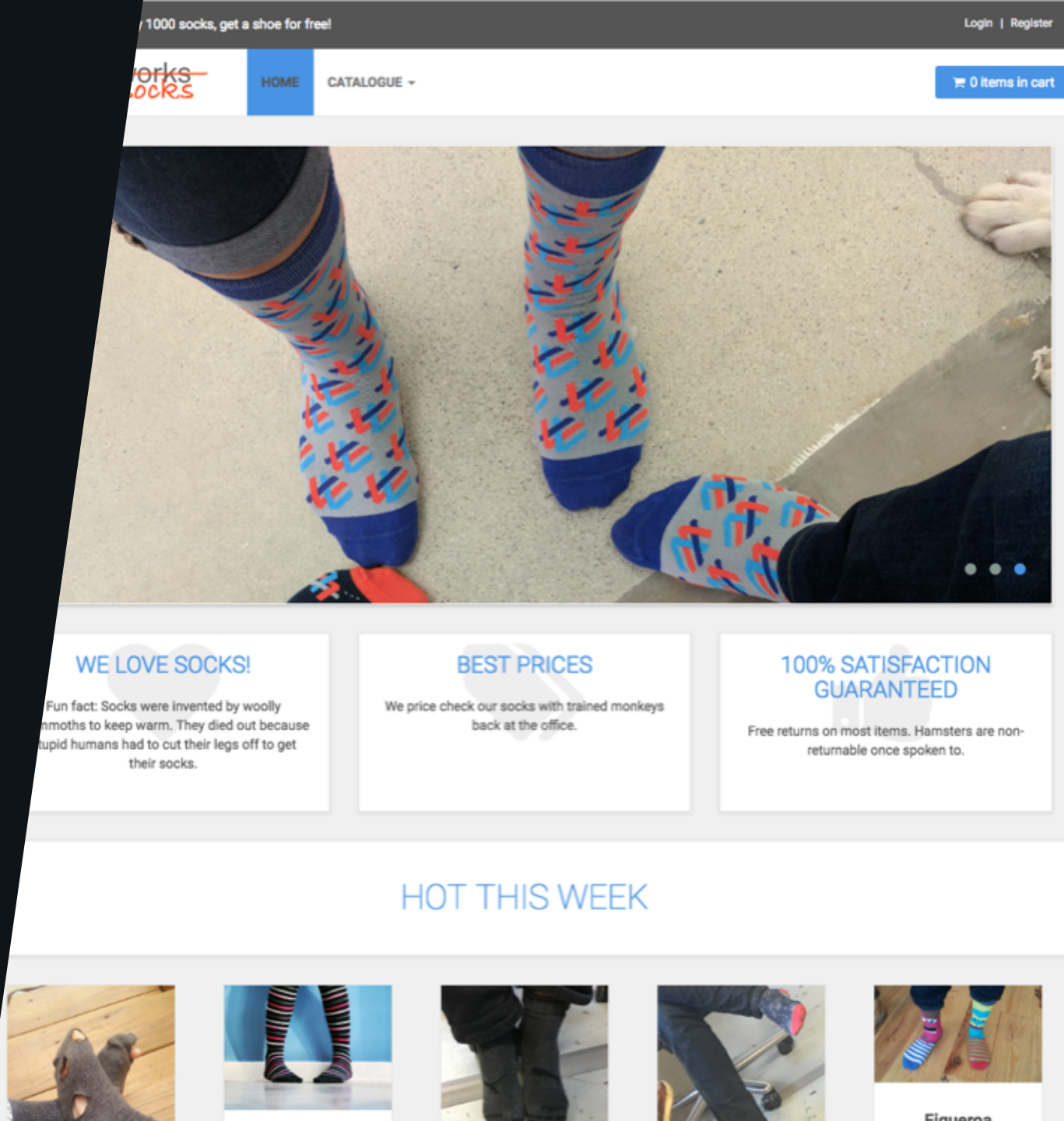


Sock Shop

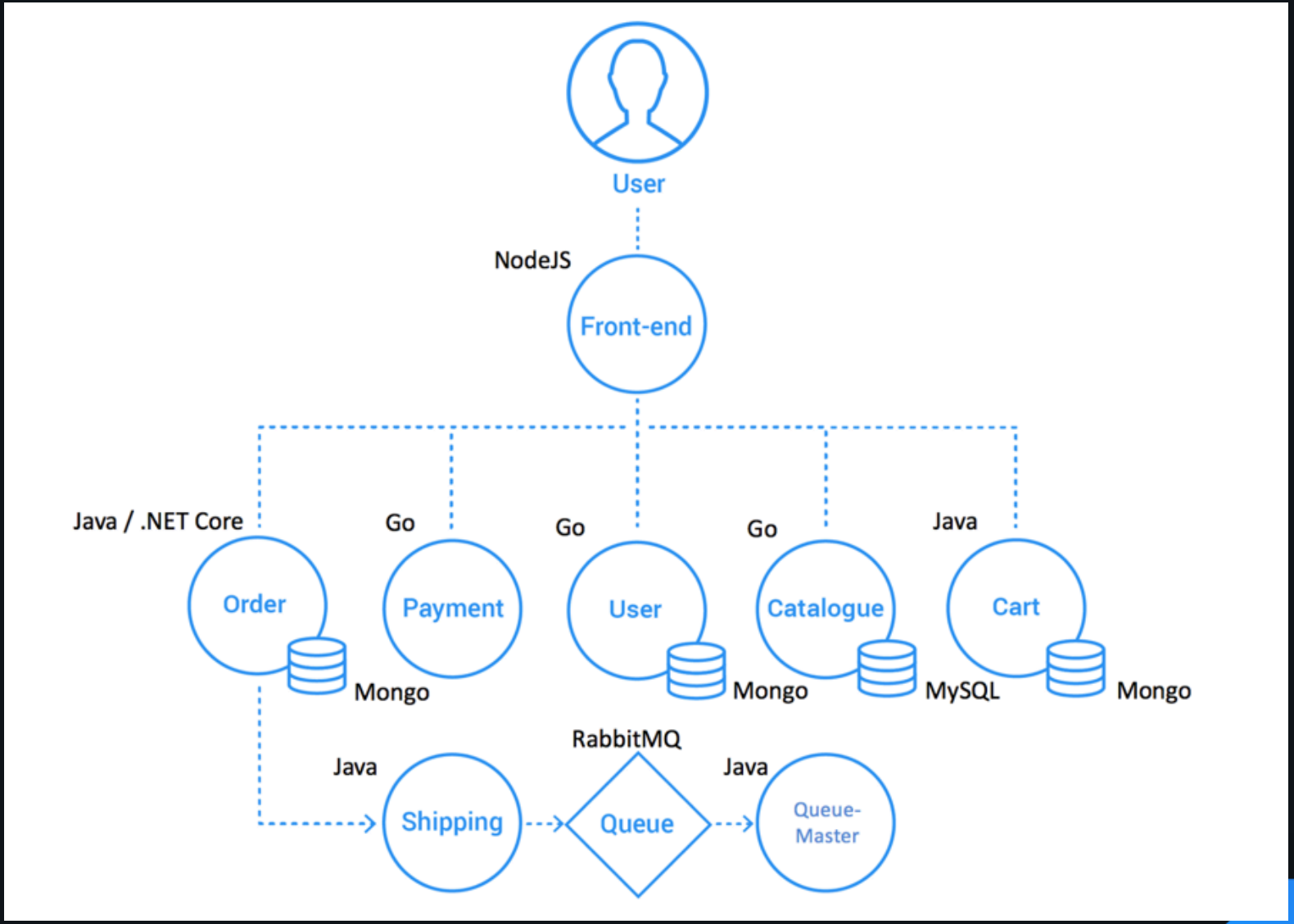

SO TELL ME AGAIN
HOW DOES ANYTHING YOUR MARKETING DEPARTMENT SAYS ACTUALLY BECOME REALITY?
memegenerator.net

Sock Shop

- Based on : http://socks.weave.works
- Open source microservice sample app
- Polyglot
  - Java Spring Boot
  - Go kit
  - Node.js
  - RabbitMQ (order queuing)
  - Mongo, MySQL (data stores)



Sock Shop architecture

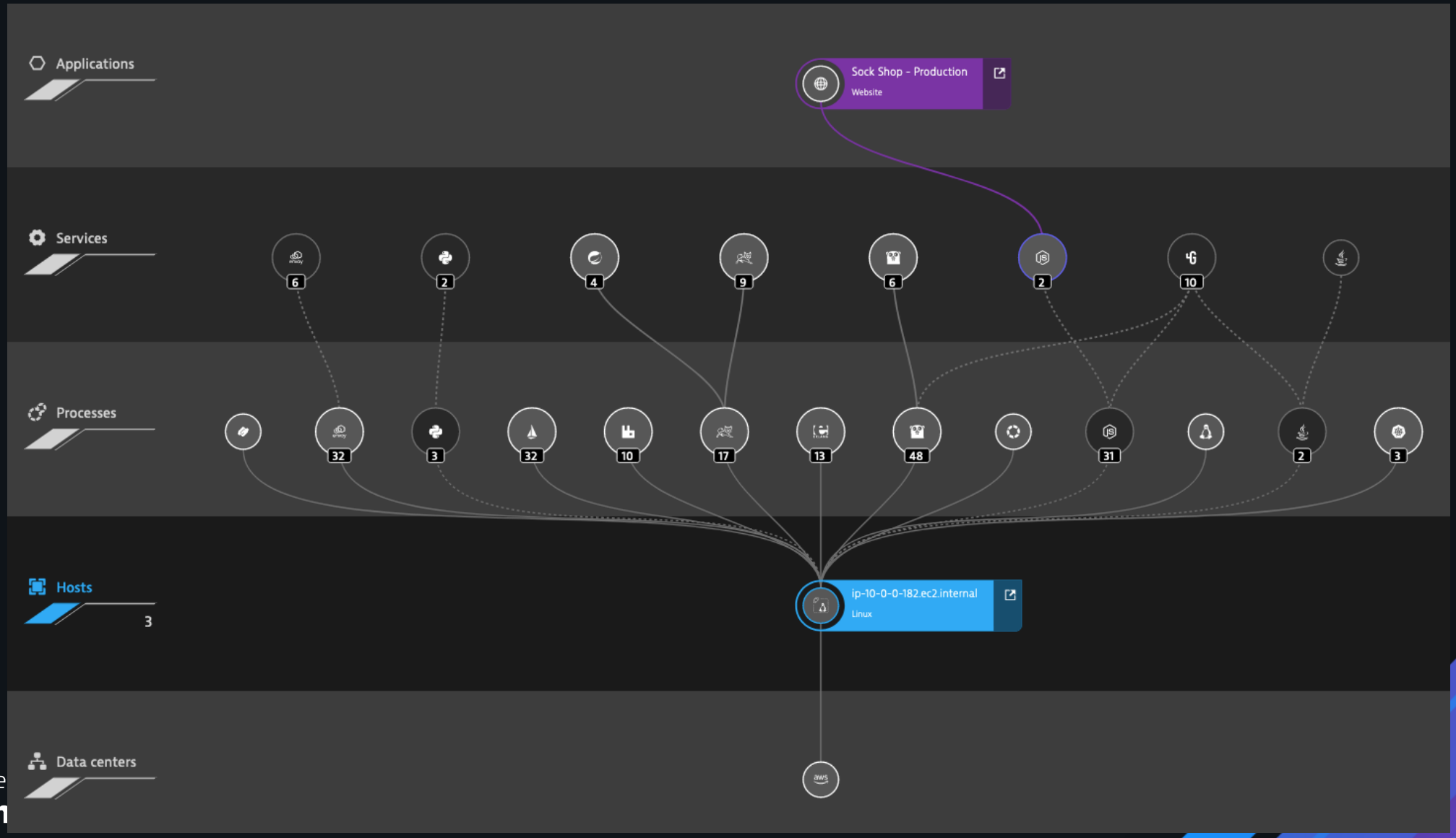


Out-of-the-box Kubernetes monitoring

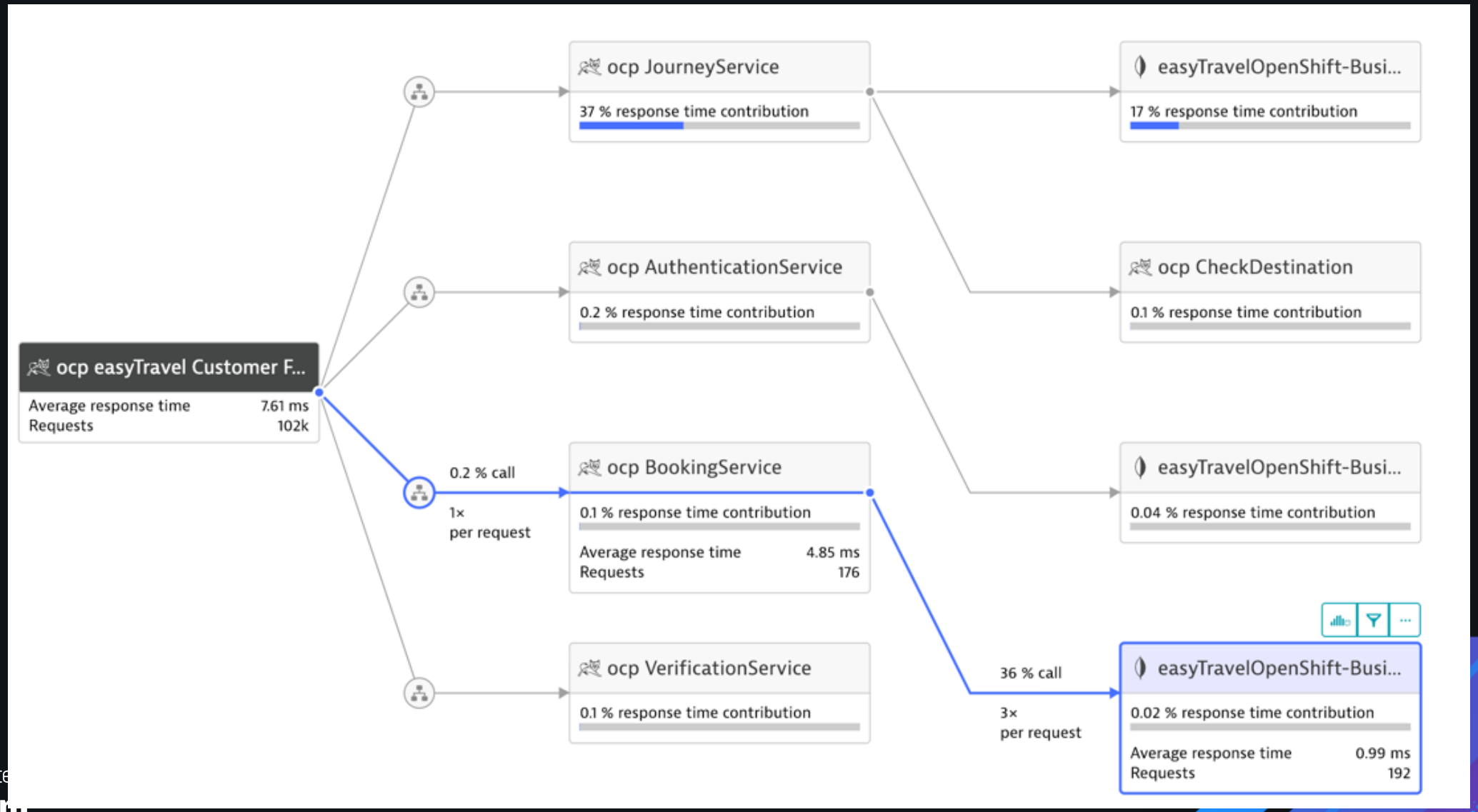
What you get ootb with Dynatrace for k8s?

- Automatic agent management by Operator
  - Every worker node have an agent deployed via daemonset
  - Automatic agent update
  - Automatic discovery : no image or container manipulation required
  - Automated full stack monitoring: hosts, containers, processes, services, application
  - Automatic capture of container and pod metadata
- Automated transaction tracing + deep dive code-level visibility
- Cluster nodes and workload monitoring
- Container monitoring

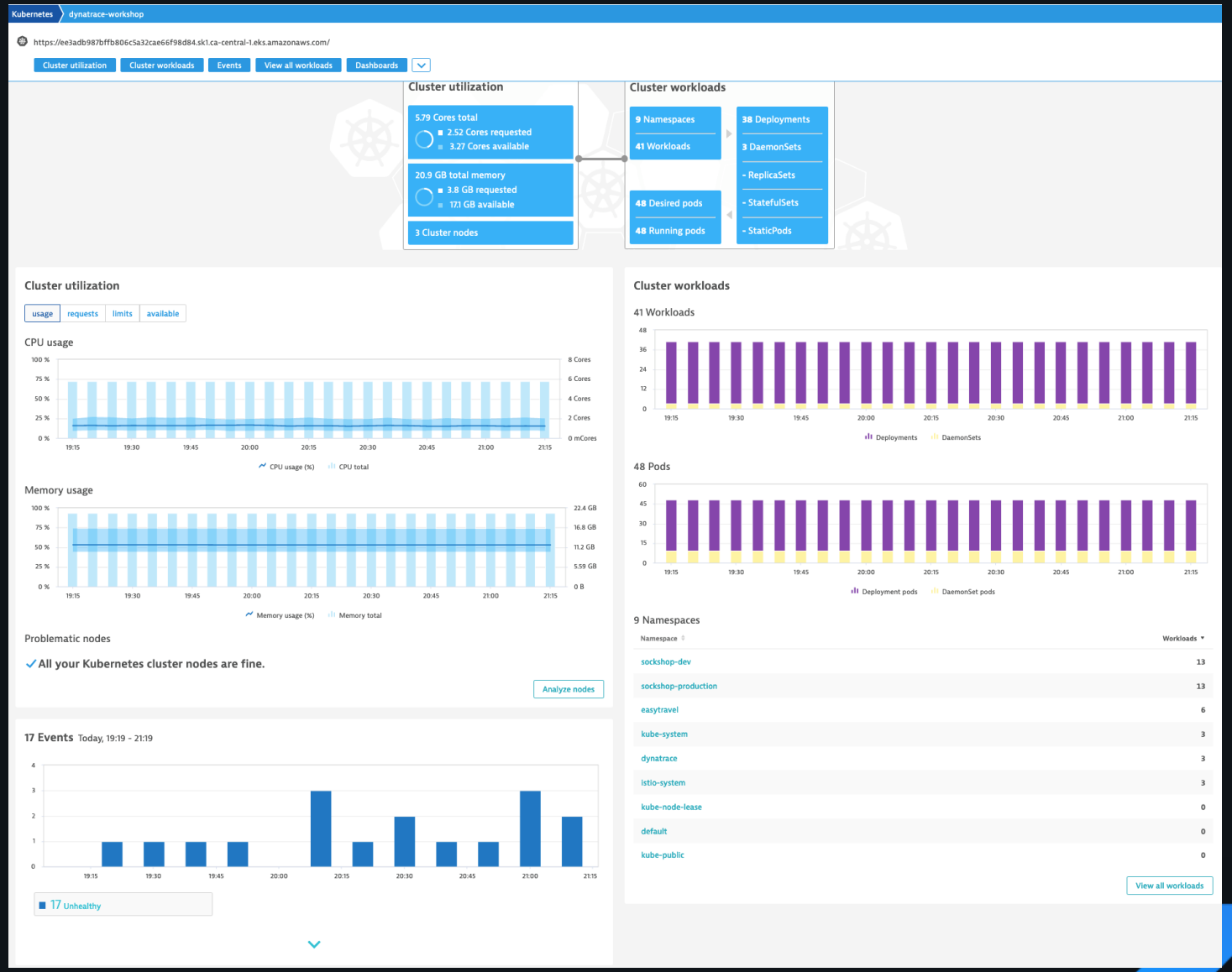
Automatic discovery and dynamic topology



Automatic distributed tracing

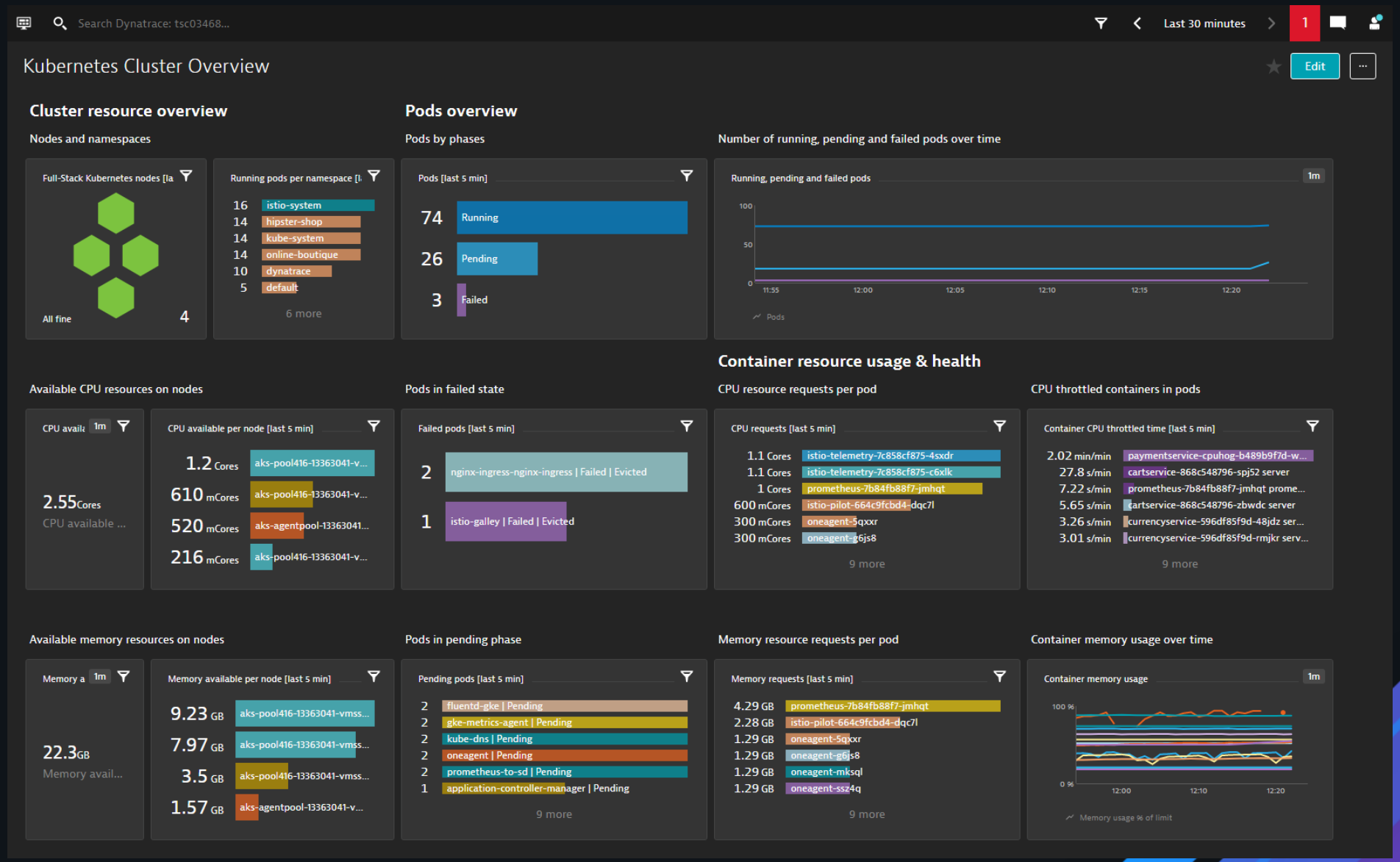


Kubernetes cluster nodes and workload monitoring



dynatrace
Perform

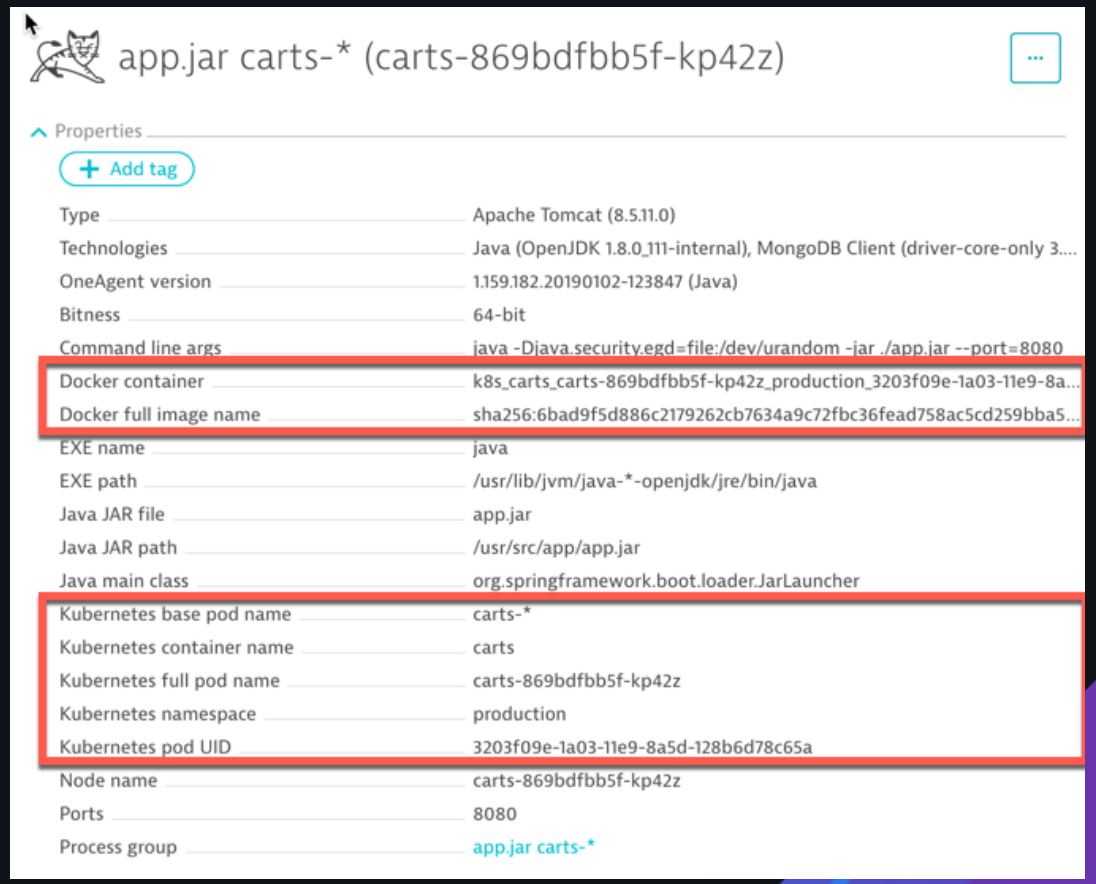
Kubernetes cluster nodes and workload monitoring



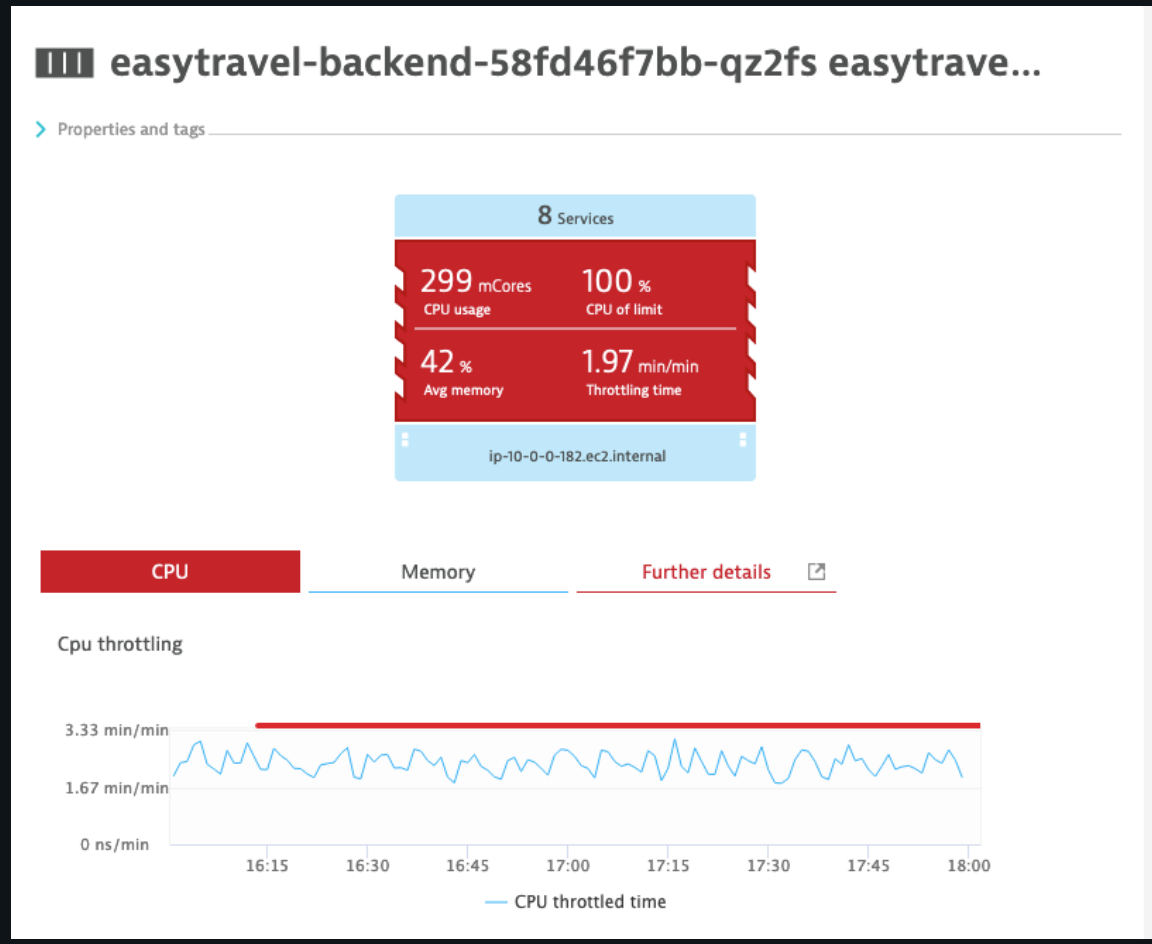
dynatrace
Perform

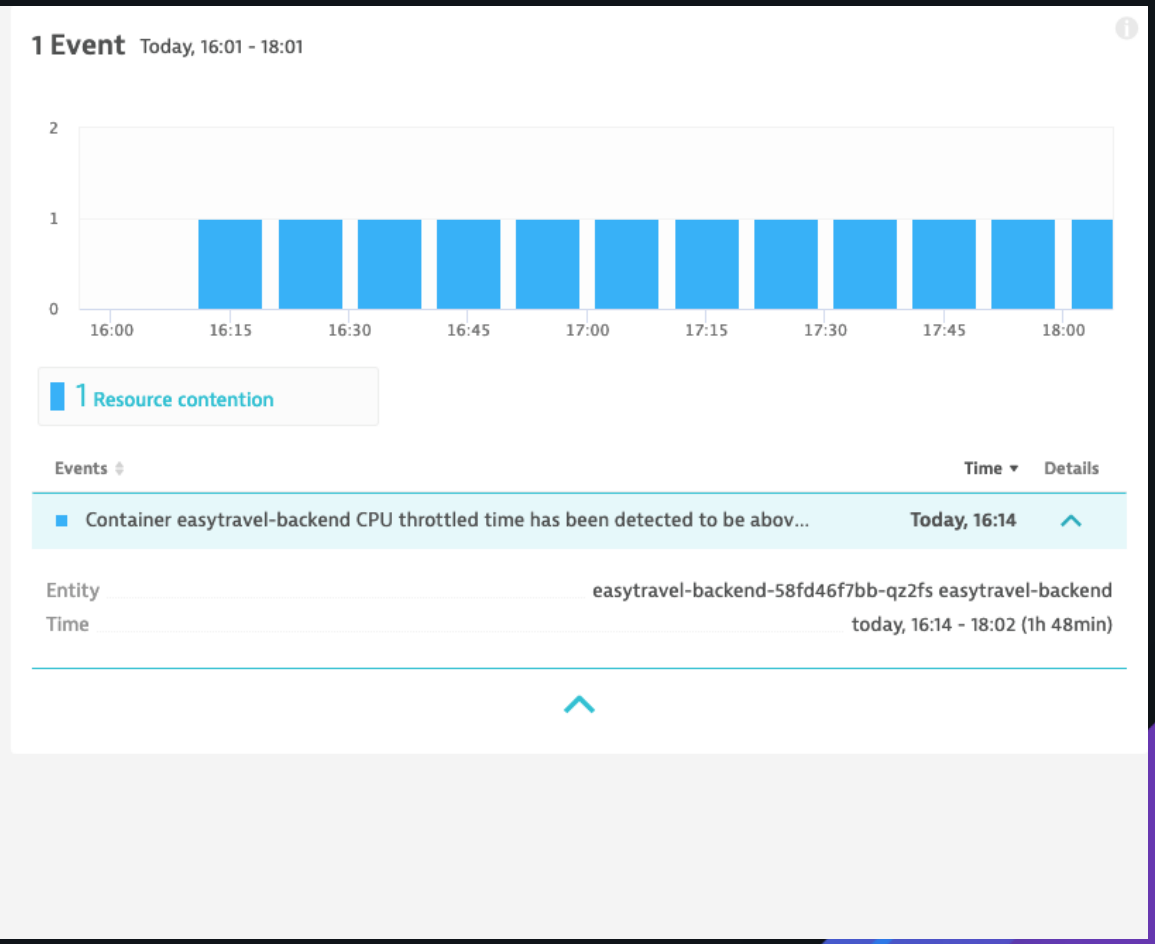
Automatic container and pod metadata

- Docker container name
- Docker container image name
- **Kubernetes base pod name**: User-provided name of the pod the container belongs to
- **Kubernetes container**: Name of the container that runs the process
- **Kubernetes full pod name**: Full name of the pod the container belongs to
- **Kubernetes namespace**: Namespace to which the containerized process is assigned
- **Kubernetes pod UID**: Unique ID of the related pod.



Automatic container metrics





Automation rules to leverage Kubernetes context

Kubernetes metadata : Labels

- Key/value pairs that are attached to objects, such as pods
- Used to specify identifying attributes of objects that are meaningful and relevant to users and organizations
- Enable users to map their own organizational structures onto system objects in a loosely coupled fashion
- Via a label selector, the client/user can identify a set of objects in Kubernetes

Kubernetes metadata : Annotations

- Used to attach arbitrary metadata to Kubernetes objects.
- Metadata can be structured or unstructured, large or small.
- Unlike labels, annotations are not used to identify or select objects

Kubernetes metadata import in Dynatrace

- Ideally, you want to avoid duplicating selectors, filters, metadata across multiple tools having those uniquely defined in Kubernetes.
- Kubernetes can be configured to allow the OneAgent to import labels and annotations in Dynatrace, avoiding duplications.
- The OneAgent will use a pod service account to query for its metadata via the Kubernetes REST API.
- Kubernetes metadata can be useful in Dynatrace for filtering, structure and providing additional context. Management Zones, Alerting Profiles, API call filters, dashboard filters.

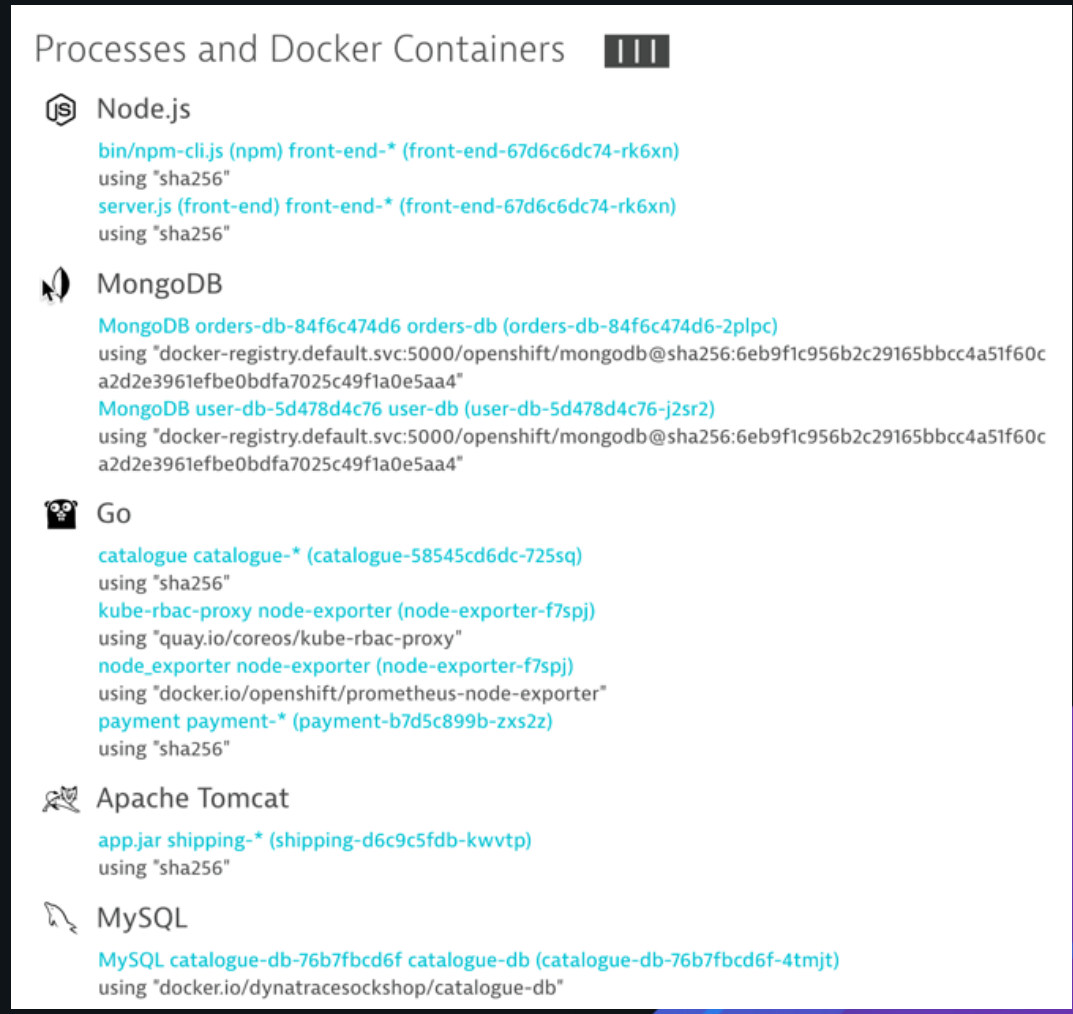
Exercise #2
Import k8s Labels & Annotations

Process Group and Service Naming

- Depending on the technology, Dynatrace will apply out-of-the-box rules in a best effort to automatically name the processes, process groups and services that are discovered.
  - In some cases, the auto-generated names may either be too generic or not reflect your naming standards
  - Also, in large enterprise multi-platform and multi-application environments, you can easily get lost without a good naming convention.

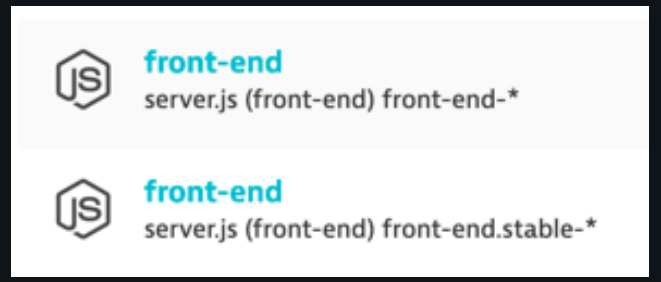
Process Group Naming

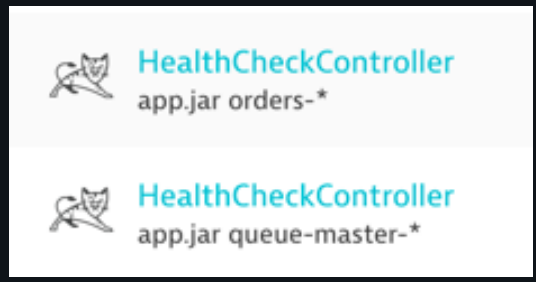
- With these process names, how do you know:
  - Which process is running in a pod on Kubernetes?
  - Which one is not, outside of Kubernetes?
  - Which one is part of the application or part of the Kubernetes platform?
  - Which namespace?
  - Where is my container name?
  - Which one runs in production or in dev?
  - Which one is my canary release vs my stable release?

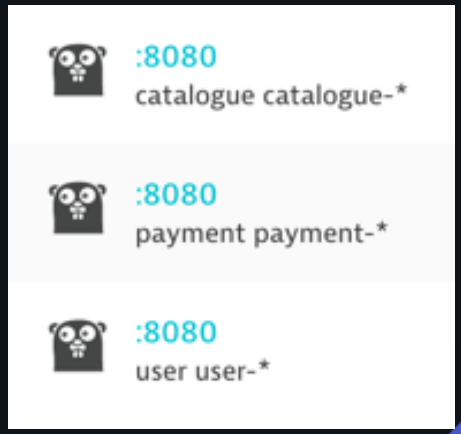


Service Naming

- Same for Services:
  - Which service is running in a pod on Kubernetes?
  - Which one is not, outside of Kubernetes?
  - Which one is production? Which one is dev?
  - I have duplicate names even though they are different services
  - What can I do with a service called ":8080"?







Customize Process Group and Service Naming

- In Dynatrace, you can define process group naming rules
  - Applied automatically, no restart needed
  - Don't affect how processes and services are discovered and grouped
  - Can be modified or deleted without any data loss
  - Naming rules can leverage process and service metadata to enforce granular and contextual naming standards
- Recommended best practice for k8s/ocp

Exercise #3
Customize Service Naming

Organize your Kubernetes monitoring entities with Management Zones

Why Management Zones?

- Kubernetes clusters are infrastructure shared by multiple delivery stages, projects or lines of business
  - Scale and complexity of the moving pieces can make it challenging to find what you want to focus on
  - You want also to be able to control access and permissions
- Management Zones allow to set up flexible and sophisticated rules filter and segregate data and access
  - Per namespace
  - Per label and/or annotation
  - Per application
  - App-level data vs infrastructure-only

Exercise #4
Play with Management Zones

Alerting profiles

Why Alerting Profiles?

- Control the delivery of problem notifications across your organization's alerting channels
- Send the relevant alerts to the right people (profile)
- For example, at HaveFun Company:
  - k8s infra team receive their alert via email
  - SockShop devops have a Slack channel
  - SockShop carts service devs have their own Slack channel
  - EasyTravel team uses JIRA

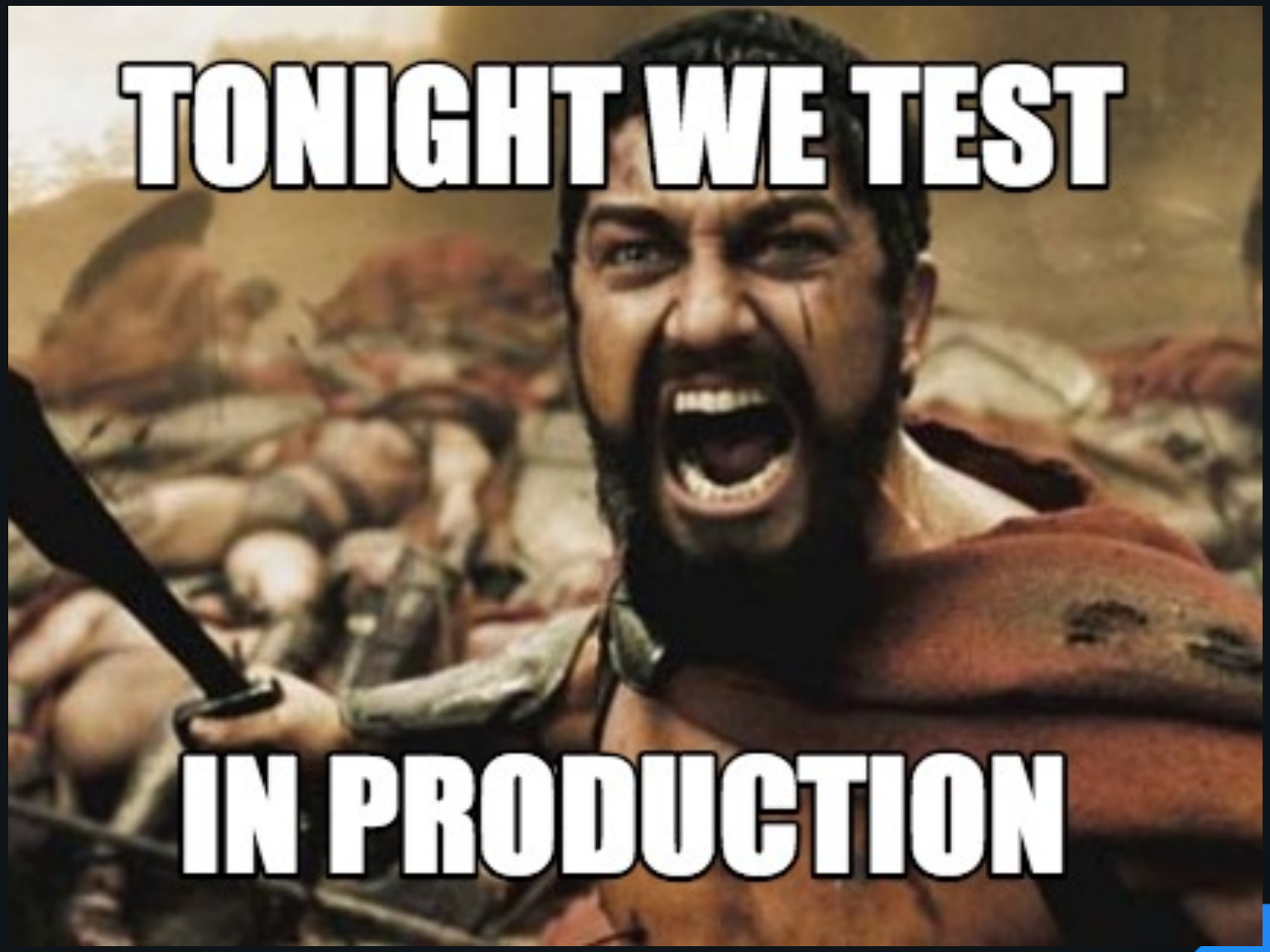
Exercise #5
Set up Alerting Profiles

New Sock Shop smart shopping cart release?

Exercise #6
Detect Performance Problems

Marketing promo campaign

Testing? Again? Not today, because...

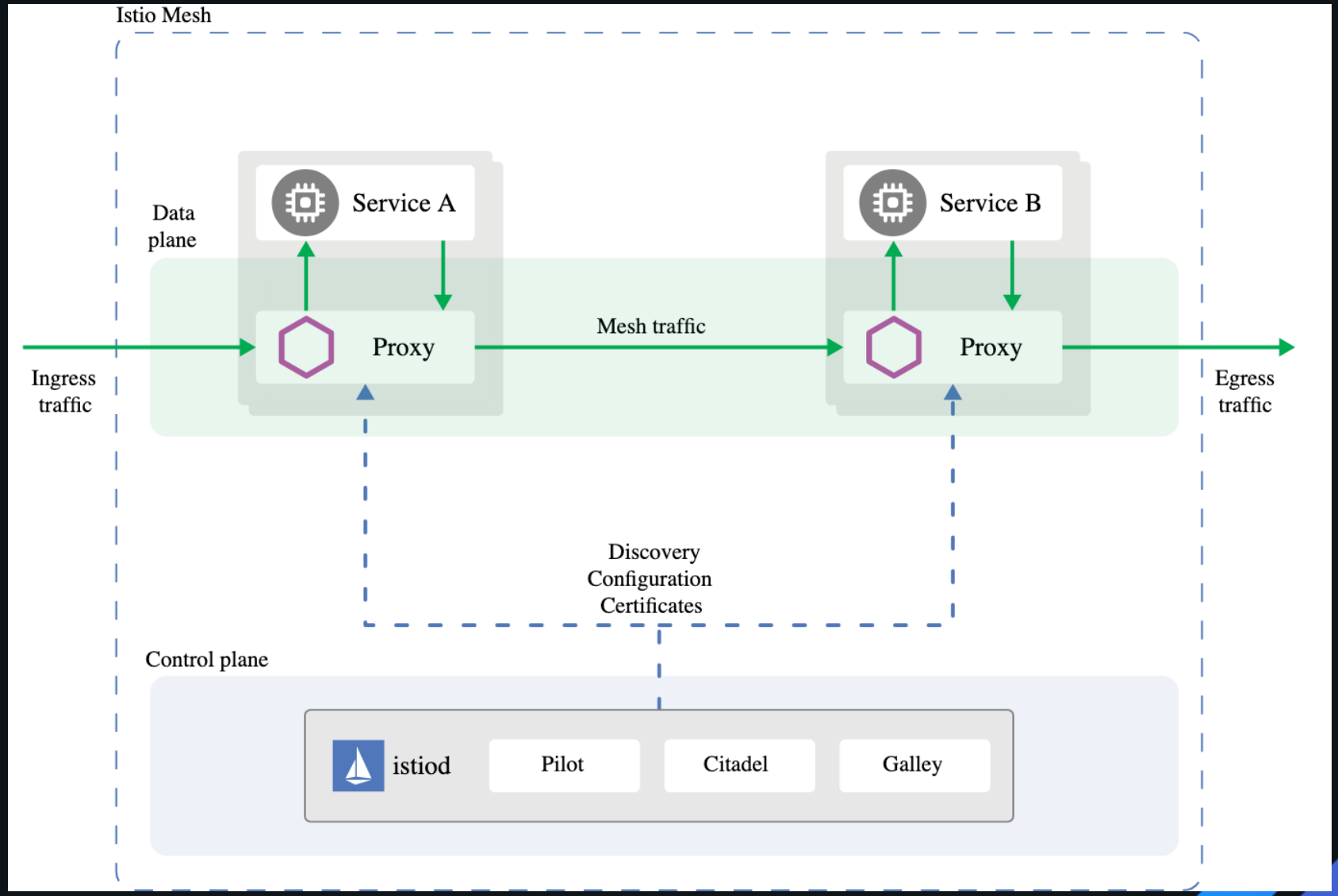


Traffic Management with Istio

What is Istio?

- Istio is a service mesh. Great… what is a service mesh? ☺
- A service mesh provides traffic monitoring, access control, discovery, security, resiliency, and other useful things to a group of services.
- Istio does all that, but it doesn't require any changes to the code of any of those services.
- Intelligent proxies (Envoy) deployed as sidecars mediating and control all network communication between microservices

Architecture



100% Traffic v1

Gateway

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: sockshop-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

VirtualService

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: sockshop
spec:
  hosts:
  - "*"
  gateways:
  - sockshop-gateway
  http:
  - route:
    - destination:
        host: front-end.production.svc.cluster.local
        subset: v1
```

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: front-end
  labels:
    app: front-end
  namespace: production
spec:
  ports:
  - name: http
    port: 8080
    targetPort: 8080
  selector:
    app: front-end
```

Deployment v1

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: front-end-v1
  namespace: production
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: front-end
        version: v1
    spec:
      containers:
      - name: front-end
        image: front-end-0.4.1
```

DestinationRule

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: front-end-destination
spec:
  host: front-end.production.svc.cluster.local
  subsets:
  - name: v1
    labels:
      version: v1
```

100% Traffic v1

Gateway

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: sockshop-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

VirtualService

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: sockshop
spec:
  hosts:
  - "*"
  gateways:
  - sockshop-gateway
  http:
  - route:
    - destination:
        host: front-end.production.svc.cluster.local
        subset: v1
```

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: front-end
  labels:
    app: front-end
  namespace: production
spec:
  ports:
  - name: http
    port: 8080
    targetPort: 8080
  selector:
    app: front-end
```

Deployment v1

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: front-end-v1
  namespace: production
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: front-end
        version: v1
    spec:
      containers:
      - name: front-end
        image: front-end-0.4.1
```

DestinationRule

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: front-end-destination
spec:
  host: front-end.production.svc.cluster.local
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
```

Deployment v2

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: front-end-v2
  namespace: production
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: front-end
        version: v2
    spec:
      containers:
      - name: front-end
        image: front-end-0.4.2
```

Traffic Distribution v1 & v2

Gateway

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: sockshop-gateway
spec:
  selector:
    istio: ingressgateway
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"
```

VirtualService

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: sockshop
spec:
  hosts:
  - "*"
  gateways:
  - sockshop-gateway
  http:
  - route:
    - destination:
        host: front-end.production.svc.cluster.local
        subset: v1
      weight: 50 #v1
    - destination:
        host: front-end.production.svc.cluster.local
        subset: v2
      weight: 50 #v2
```

Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: front-end
  labels:
    app: front-end
  namespace: production
spec:
  ports:
  - name: http
    port: 8080
    targetPort: 8080
  selector:
    app: front-end
```

Deployment v1

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: front-end-v1
  namespace: production
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: front-end
        version: v1
    spec:
      containers:
      - name: front-end
        image: front-end-0.4.1
```

DestinationRule

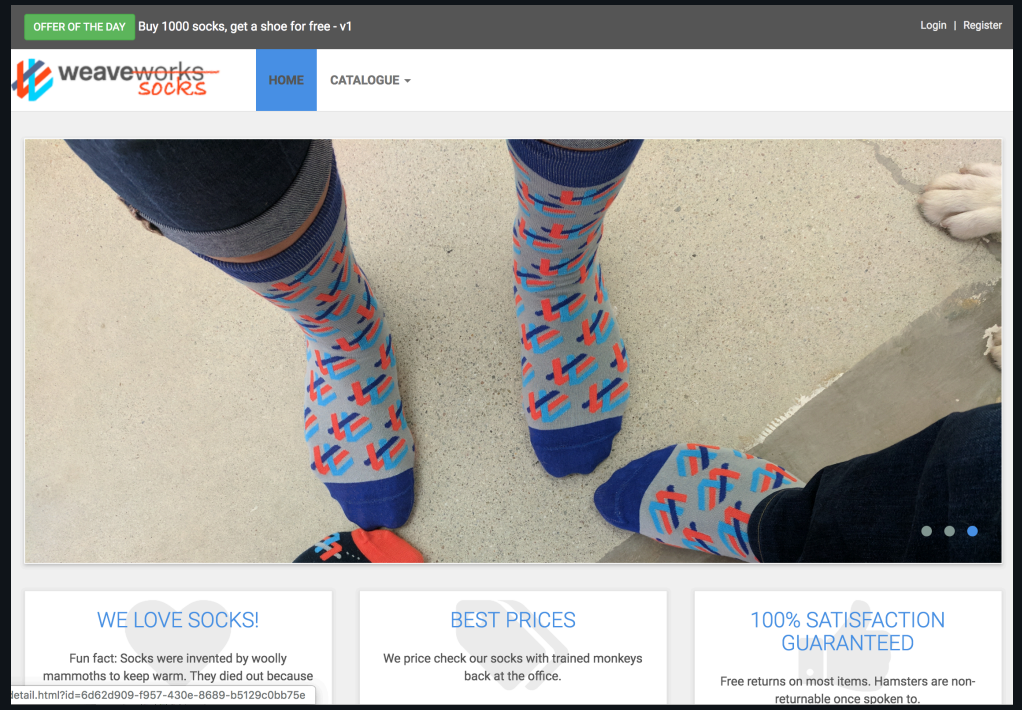
```yaml
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: front-end-destination
spec:
  host: front-end.production.svc.cluster.local
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
```

Deployment v2

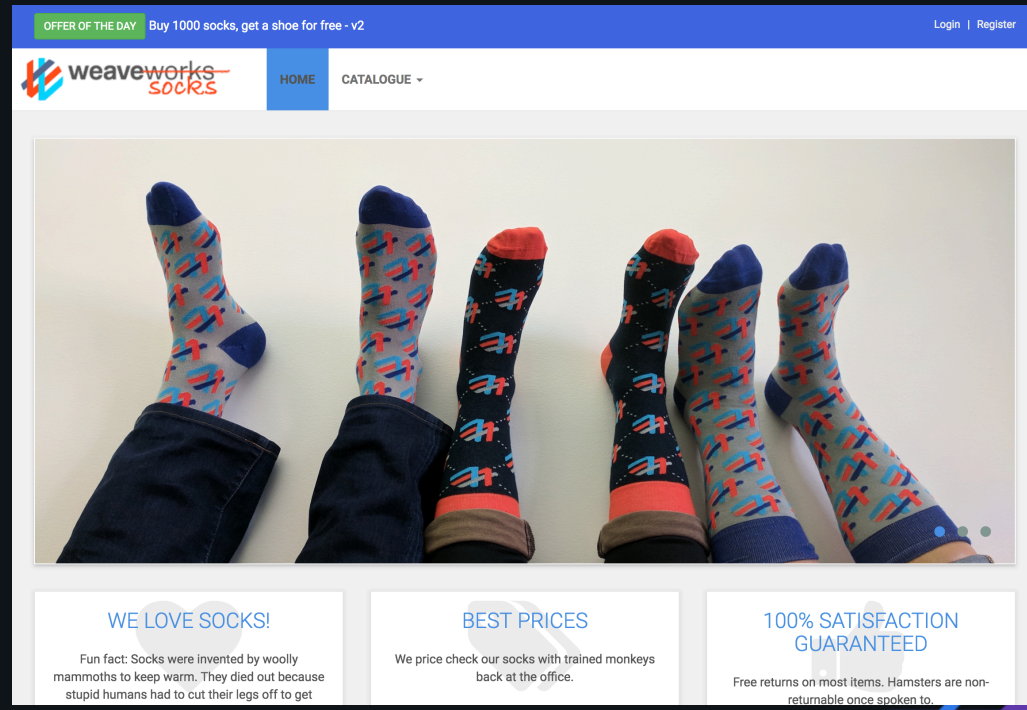
```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: front-end-v2
  namespace: production
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: front-end
        version: v2
    spec:
      containers:
      - name: front-end
        image: front-end-0.4.2
```

Front-end : difference between v1 and v2

v1



v2

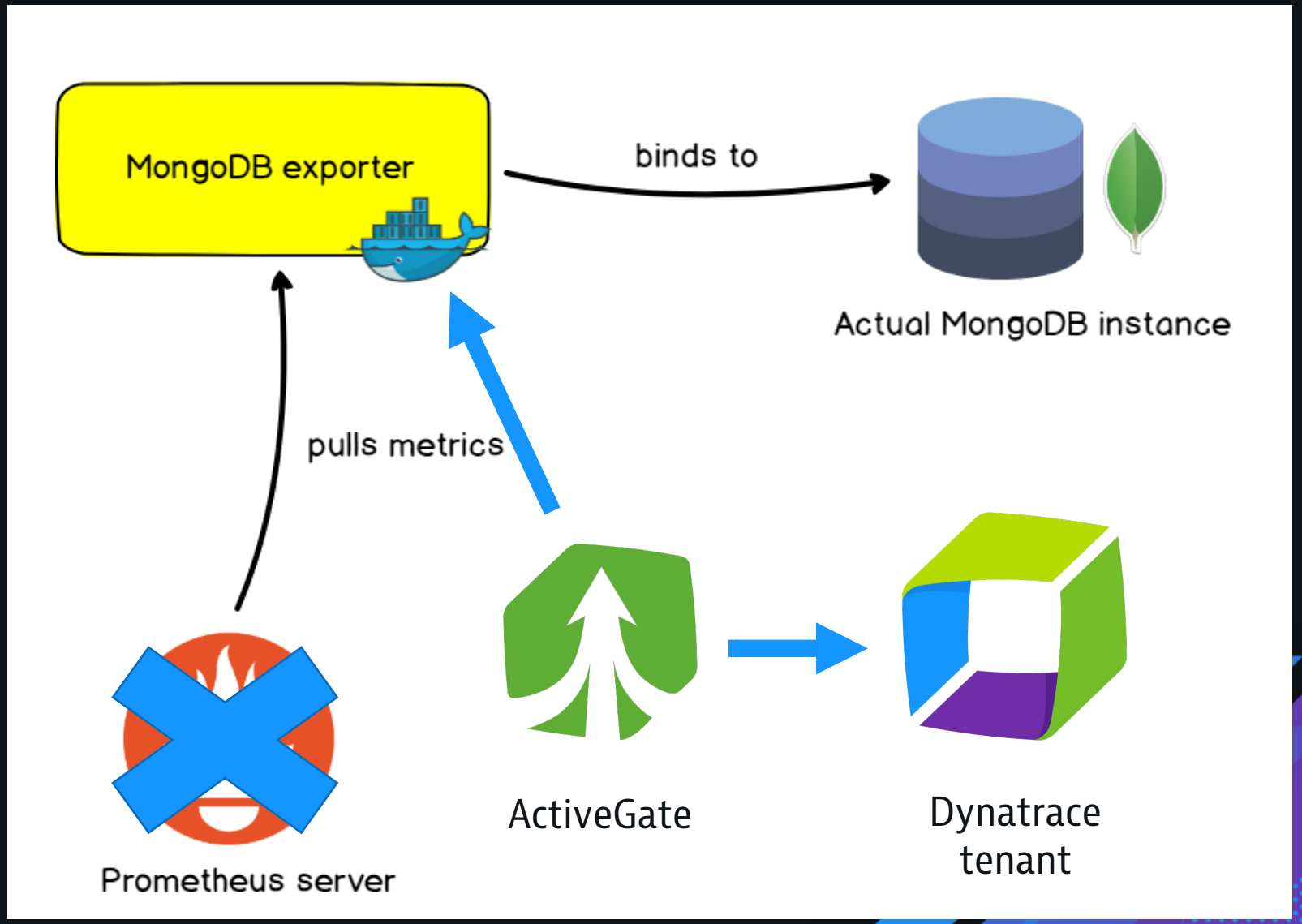


Exercise #7
Deploy a Canary

Prometheus metrics import

Prometheus metrics import

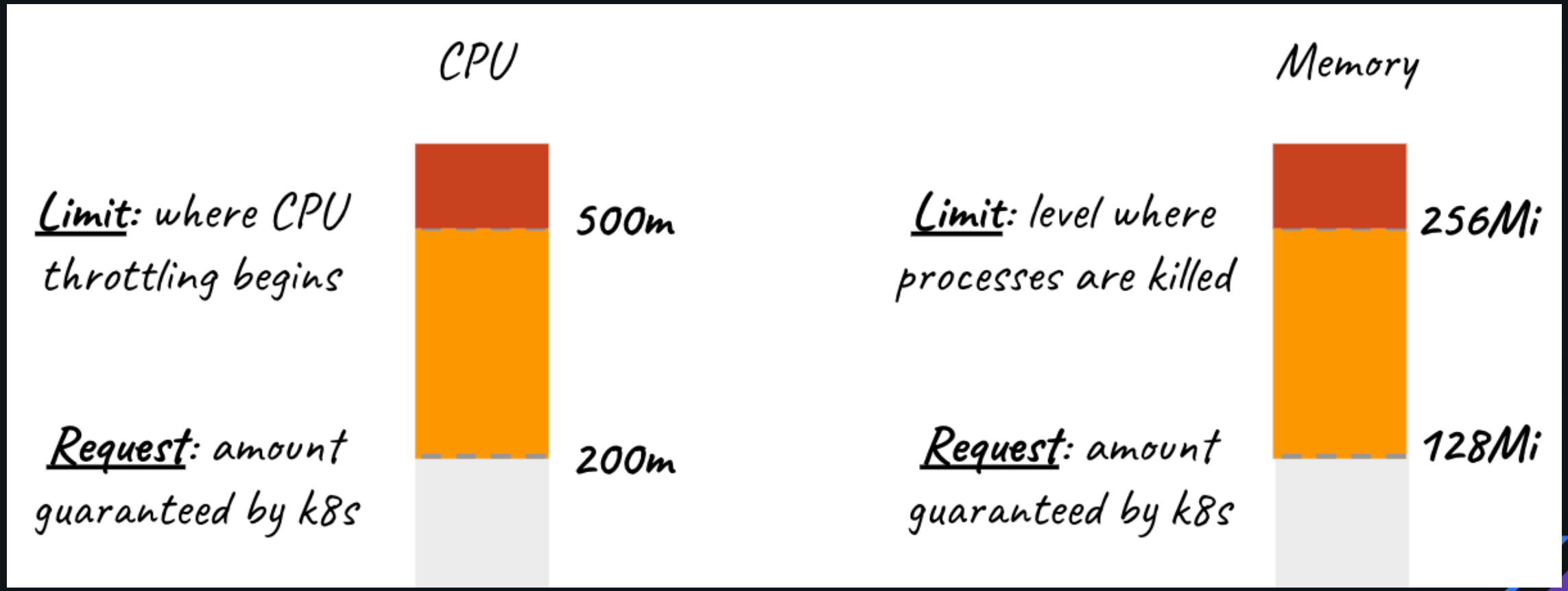
- Support Prometheus/Open Metrics format
- ActiveGate scrapes from exporter endpoints
- Metrics become first class citizen in Dynatrace



Exercise #8
Import Prometheus Metrics

K8s workload resource management

Container compute resources



credit: http://blog.kubecost.com/blog/requests-and-limits/

Exercise #9
Manage your Workload Resource Usage

What’s coming?

New logs support for Kubernetes

Foward logs to Dynatrace

Rest ENDPOINT
https://<env-URL>/api/v2/logs/ingest

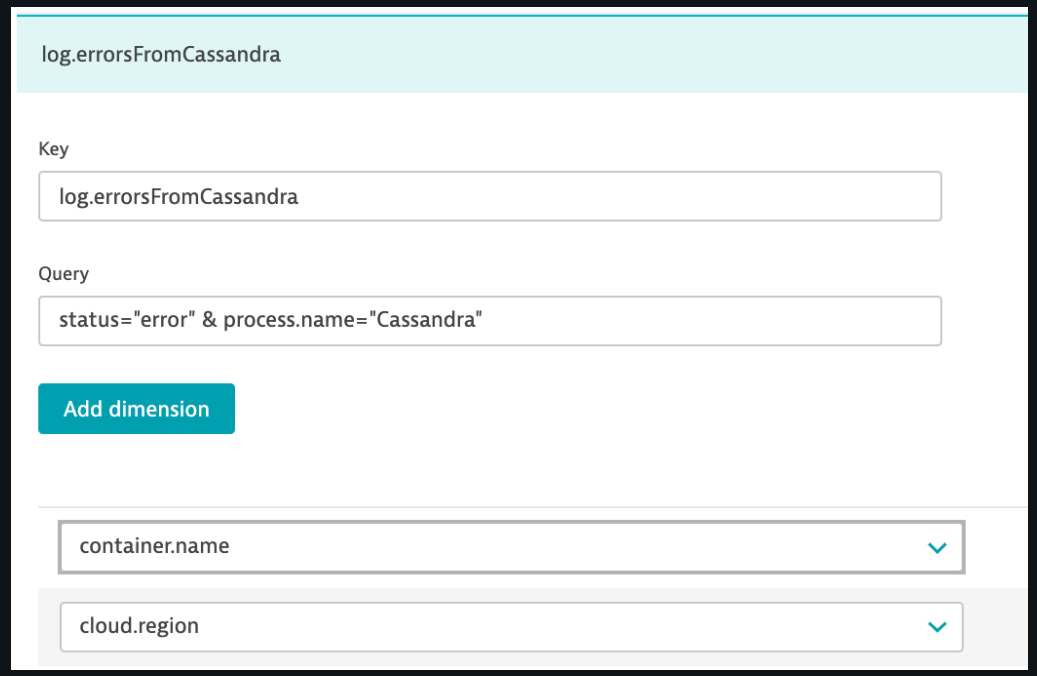
Configures

Dynatrace Operator

Log and Kubernetes events alerting

Metrics subscription based on log & events streams.

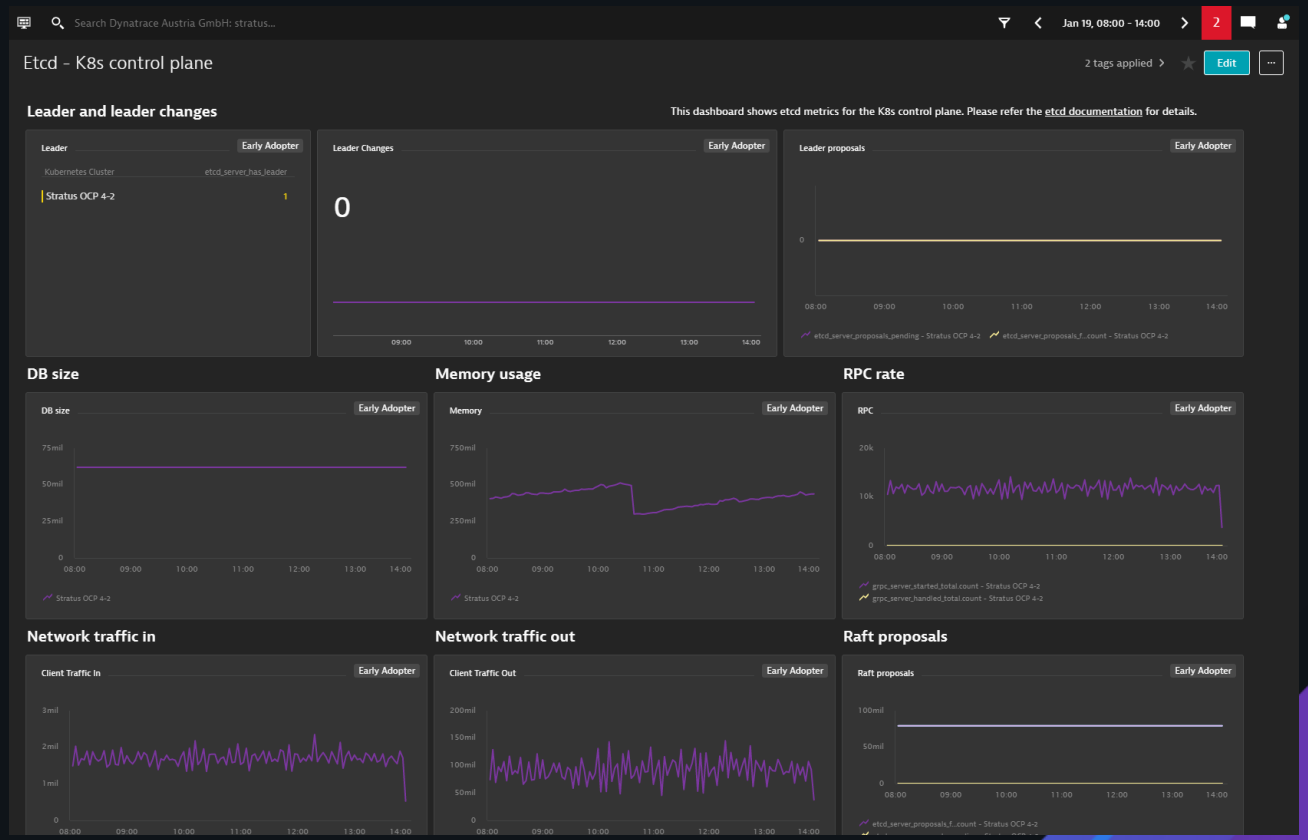
- Empower DAVIS AI with insights from log streams, root cause analysis
- Flexible alerting based on contextual event and log data

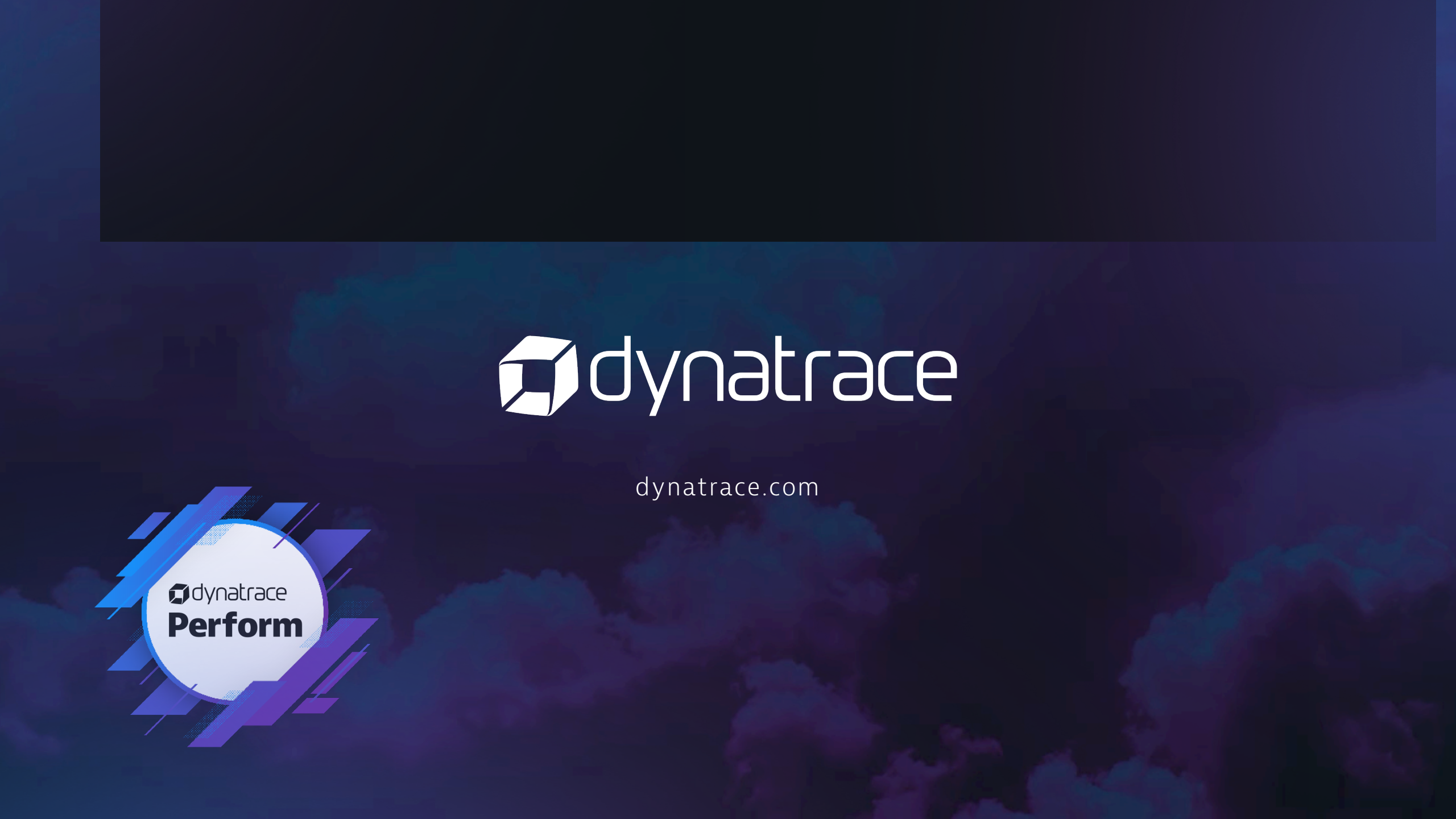


Control plane and others

- Understand control plane health
- etcd, api-server, controller manager, kubelet,…
- Dashboards and alerts

- Istio, LinkerD, Knative…
- Persistent volumes




dynatrace
dynatrace.com
dynatrace
Perform