PhonePe Open Source

Drove

Distributed Container Orchestrator

Vishnu Naini
Site Reliability Engineer, PhonePe

Santanu Sinha
Chief Architect, PhonePe

PhonePe

Drove - Distributed Container Orchestrator

PhonePe Open Source

Simple,Performant, Operations-friendly

- Simple container orchestration
- Orchestrate applications and tasks
- Built to work with docker and podman container engines
- Epoch - time based task scheduler
- Drove Gateway - Route traffic to applications using NGINX
- CoreDNS plugin - Route traffic using DNS routing
- Logical successor to Apache Mesos/Marathon

For further resources visit

https://phonepe.github.io/drove-orchestrator/

drove

Why?

- Mesos - End of Life
- Highly available and fault tolerant
- Efficient resource utilization
- Simple compliance and security models
- Ease of use and management at scale
- Container centric architecture

drove

Before

- Mesos
  - End of Life
  - Not container centric
  - Too many extensions to support cgroups, docker, health checks etc.
- Traffic routing
  - Challenges at scale - Config reloads, Resource utilization, Complex routing etc.
  - Resource wastage due to layers of abstraction
- Control-plane security/authn/authz
- use-case specific tools - e.g. Service Discovery / Traffic shaping / Disaster recovery
- Complicated and disruptive maintenance and upgrade procedure

drove

At scale

- Leverage existing and powerful external services (e.g., service discovery, auto-scaling, key-value storage) for scalability and simplicity.
- Simplified network architecture for operability
- Utilize well-understood platforms.
    - Nginx for gateway
    - Docker for resources/networking/deployment

Powering compute @ PhonePe

- Tested at scale with over 20000+ cores, 300+ executors, and 2000+ applications per cluster
- Proven at scale with millions of requests per second across 200+ clusters, 3500+ executors, and 320k+ cores over 2 years
- Running on across diverse environments including on-premise bare metal, virtual machines, and public clouds
- Multiple operating systems and CPU architectures

drove

Workloads

- **Applications** - A representation of an application version running on the cluster
    - **Instances** - Each container running on the cluster that belongs to an application
- **Tasks** - A transient task/job on the cluster which runs only for some time unlike applications
- **Local services*** - Containers running on all nodes providing specialized functionality

*Coming soon

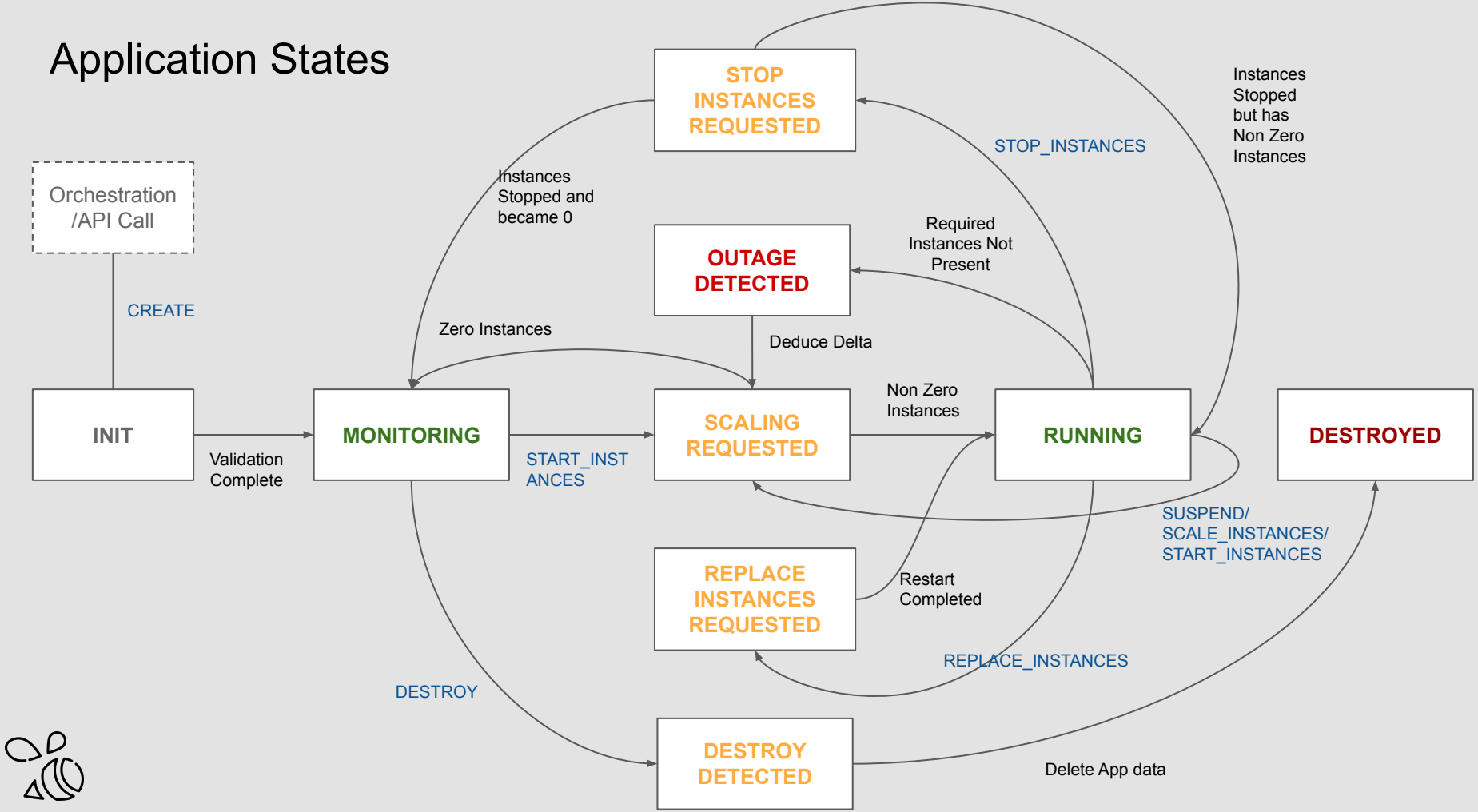
drove

Application

A representation of a service or a topology running on the cluster

- Defined by Application Specification
  - Name and Version
  - Docker coordinates, Ports, volume mounts
  - Local/RSyslog Logging
  - Readiness checks, Health checks, Pre-shutdown hooks
  - CPU/Memory requirements, Placement Policy
  - Exposure Spec - vhost, port etc
- Operations
  - CREATE, DESTROY
  - START_INSTANCES, STOP_INSTANCES, REPLACE_INSTANCES

Application States



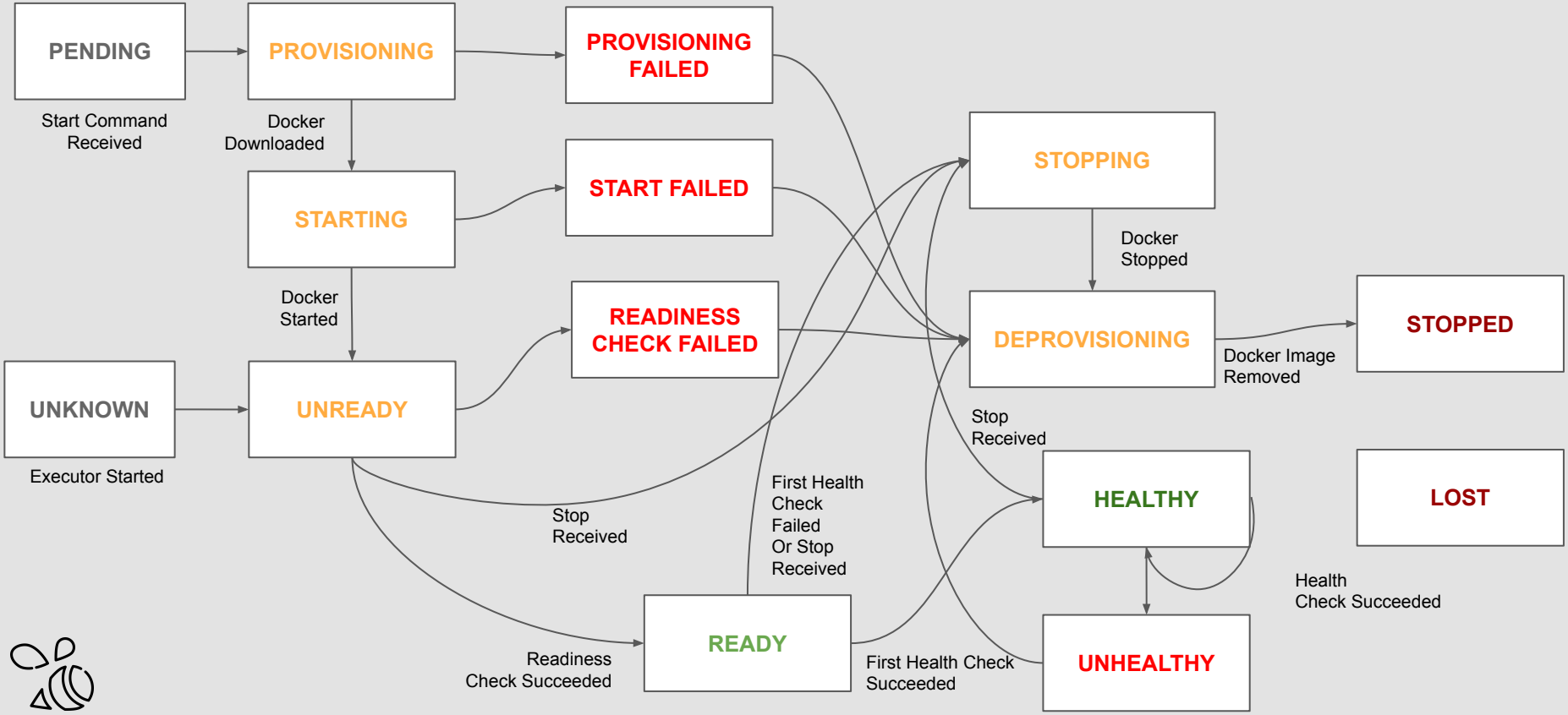
Application Instances

A running instance of an application on an executor host

- Actual representation of a running application container
- Mandatory Readiness checks and Health-checks
- Pre-shutdown hooks
- Placement policies
- Config injection
- Volume mounting
- Device mounting

drove

Instance States



Demo

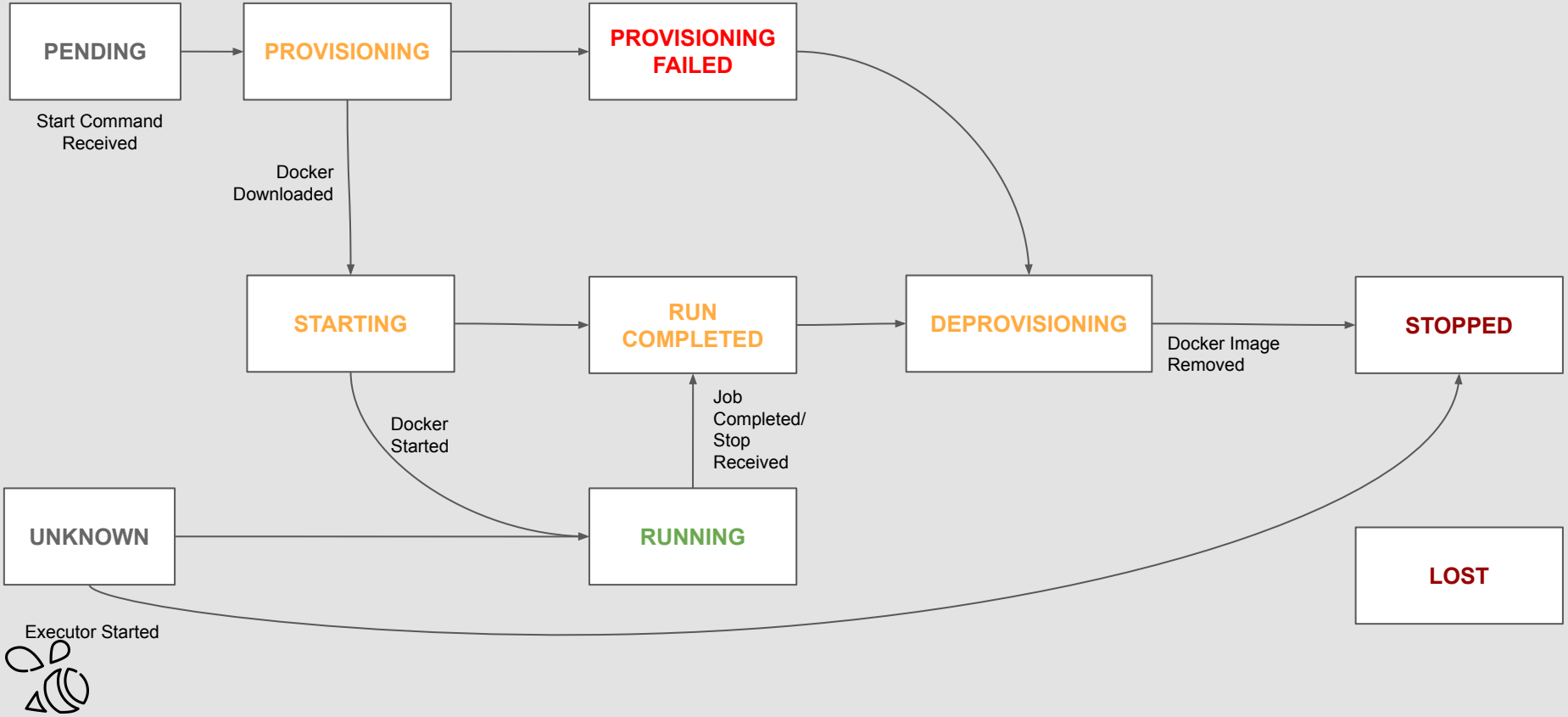
drove

Tasks

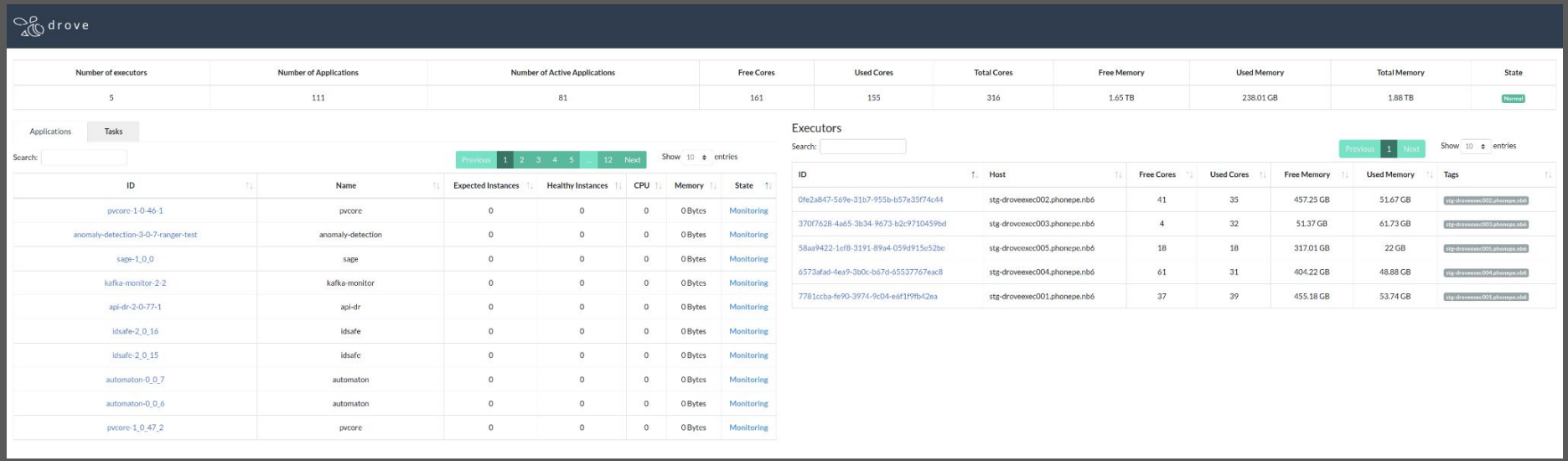
Represents a transient container execution e.g. a scheduled job

- Tasks are supposed to be spawned by apps running on drove cluster
- Task Spec
  - (Spawning/Source) App Name and Task ID
  - Docker coordinates, Ports, volume mounts
  - Local/RSyslog Logging
  - CPU/Memory requirements, Placement Policy

Task States


PENDING
PROVISIONING
PROVISIONING FAILED
Start Command Received
Docker Downloaded
STARTING
RUN COMPLETED
DEPROVISIONING
Docker Image Removed
STOPPED
Docker Started
Job Completed/ Stop Received
UNKNOWN
RUNNING
LOST
Executor Started

UI


drove
Number of executors
Number of Applications
Number of Active Applications
Free Cores
Used Cores
Total Cores
Free Memory
Used Memory
Total Memory
State
5
111
81
161
155
316
1.65 TB
238.01 GB
1.88 TB
Normal
Applications
Tasks
Search:
Previous 1 2 3 4 5 ... 12 Next
Show 10 entries
ID
Name
Expected Instances
Healthy Instances
CPU
Memory
State
pvcore-1-0-46-1 pvcore 0 0 0 0 Bytes Monitoring
anomaly-detection-3-0-7-ranger-test anomaly-detection 0 0 0 0 Bytes Monitoring
sage-1_0_0 sage 0 0 0 0 Bytes Monitoring
kafka-monitor-2-2 kafka-monitor 0 0 0 0 Bytes Monitoring
api-dr-2-0-77-1 api-dr 0 0 0 0 Bytes Monitoring
idsafe-2_0_16 idsafe 0 0 0 0 Bytes Monitoring
idsafe-2_0_15 idsafe 0 0 0 0 Bytes Monitoring
automaton-0_0_7 automaton 0 0 0 0 Bytes Monitoring
automaton-0_0_6 automaton 0 0 0 0 Bytes Monitoring
pvcore-1_0_47_2 pvcore 0 0 0 0 Bytes Monitoring
Executors
Search:
Previous 1 Next
Show 10 entries
ID
Host
Free Cores
Used Cores
Free Memory
Used Memory
Tags
0fe2a847-569e-31b7-955b-b57e35f74c44 stg-droveexec002.phonepe.nb6 41 35 457.25 GB 51.67 GB
370f7628-4a65-3b34-9673-b2c9710459bd stg-droveexec003.phonepe.nb6 4 32 51.37 GB 61.73 GB
58aa9422-1ef8-3191-89a4-059d915e52be stg-droveexec005.phonepe.nb6 18 18 317.01 GB 22 GB
6573afad-4ea9-3b0c-b67d-65537767eac8 stg-droveexec004.phonepe.nb6 61 31 404.22 GB 48.88 GB
7781ccba-fe90-3974-9c04-e6f1f9fb42ea stg-droveexec001.phonepe.nb6 37 39 455.18 GB 53.74 GB

drove

Architecture

Design Philosophy

- Focus on container performance
  - NUMA aware scheduler for CPU and Memory
  - Core pinning of container
  - Keep network layer unhindered
- Portability
  - Needs to support docker (for debian/ubuntu)
  - Podman (for CentOS etc)
- Easy maintenance
  - Should not affect running containers
  - Fast reliable upgrades
- Stability and reliability
  - Simple and fast recovery procedure
  - Containers should continue to run even if all components fail

drove

Controller

Control system for the cluster

- Execution and Orchestration
- Resource allocation and Container Placement
- Lifecycle management of Deployables
- Executor blacklisting and instance migration
- Expose UI and HTTP APIs
- Streaming logs from executors to frontend
- Stream cluster events like app deployment etc and provide topology information
- Uses ZK as data store and for leader election etc
- Highly Available with a single leader active at a time

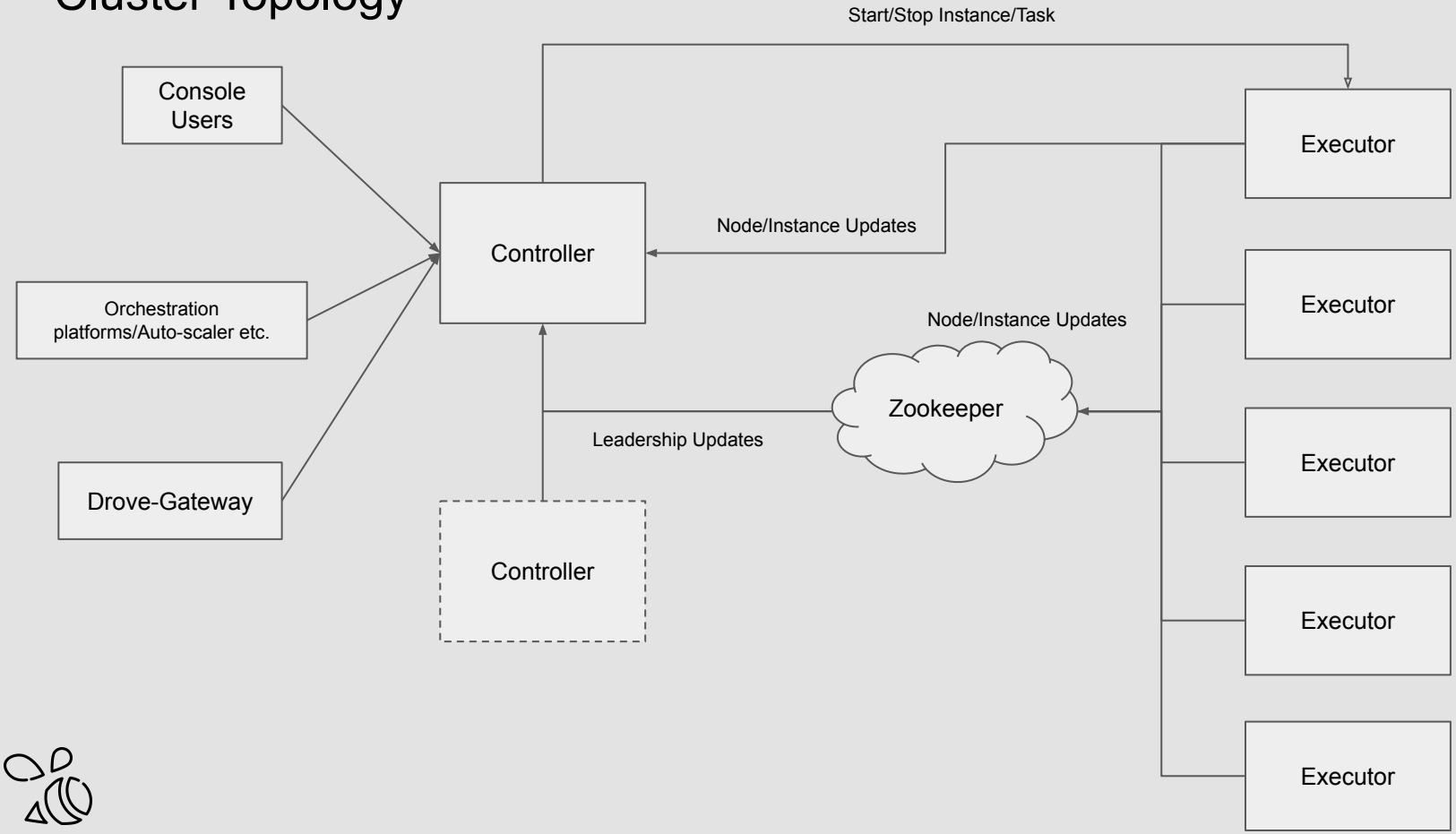
drove

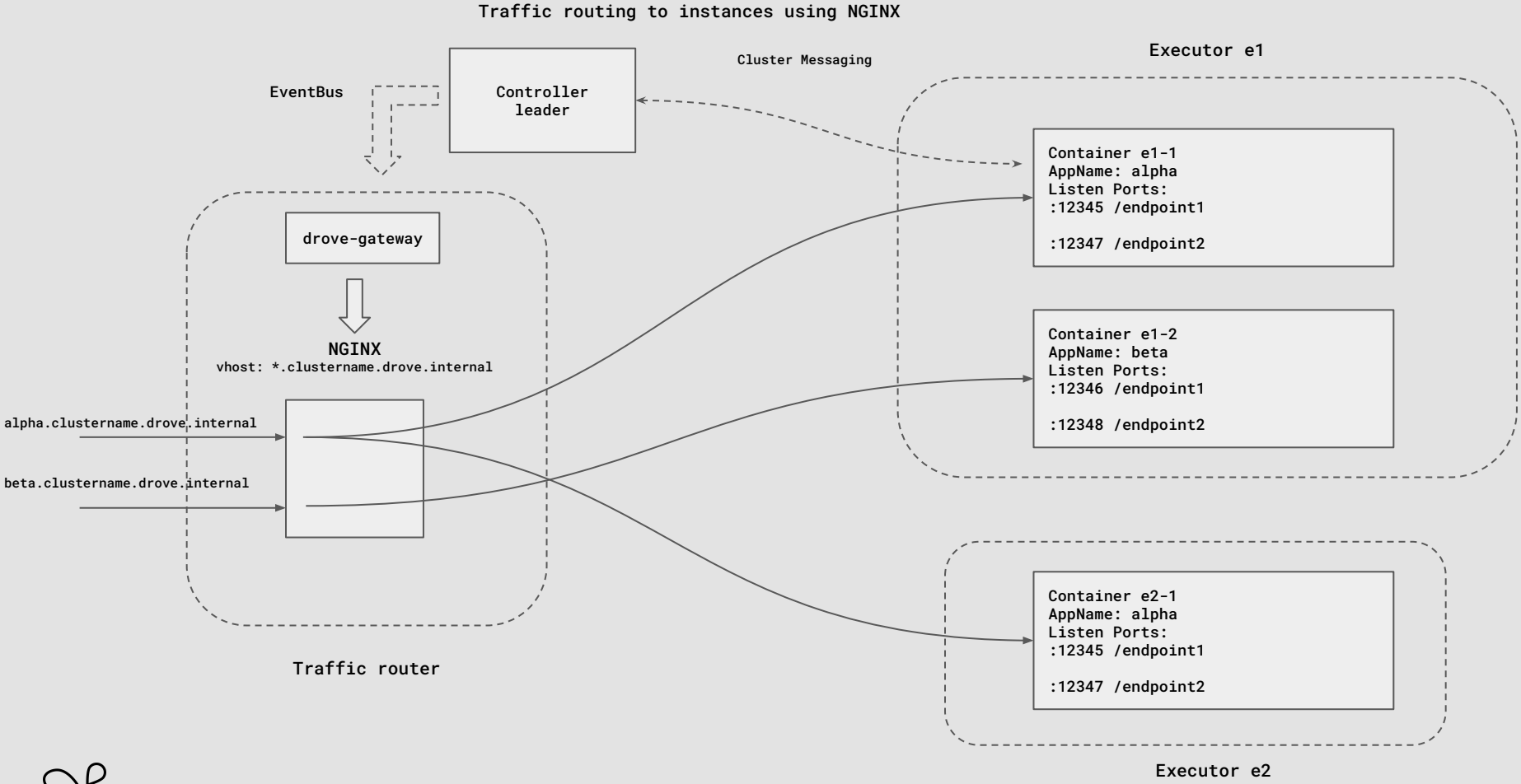
Executors

Execute Application Instances, Tasks etc

- Deployable Instance lifecycle management
- Readiness check, health check, pre-shutdown hooks execution
- Identify and kill any runaway containers that are not supposed to exist
- Exposes NUMA topology for scheduling
- Executors can be tagged based on property (e.g. has_gpu, big_mem)
    - Tags can be used in placement policies

Cluster Topology




Traffic routing to instances using NGINX
EventBus
Controller leader
Cluster Messaging
Executor e1
drove-gateway
NGINX
vhost: *.clustername.drove.internal
alpha.clustername.drove.internal
beta.clustername.drove.internal
Container e1-1
AppName: alpha
Listen Ports:
:12345 /endpoint1
:12347 /endpoint2
Container e1-2
AppName: beta
Listen Ports:
:12346 /endpoint1
:12348 /endpoint2
Container e2-1
AppName: alpha
Listen Ports:
:12345 /endpoint1
:12347 /endpoint2
Traffic router
Executor e2

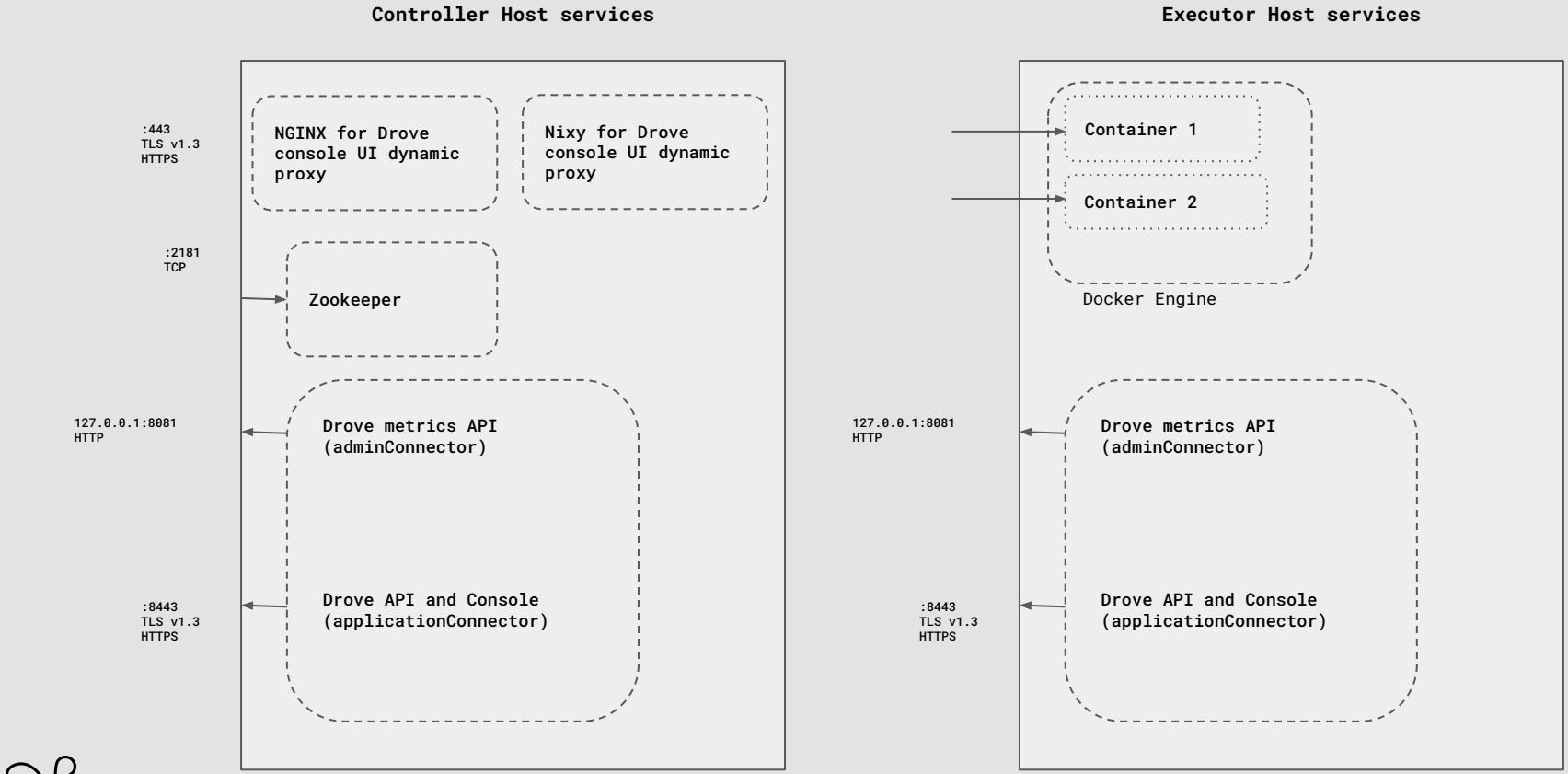
How It’s Used…

Deployment

- Debian/RPM/Docker packaging
- Controller, ZK run on the controller nodes
- Typically 3 x controllers, required number of executors and gateway nodes
- osCores reservation using systemd CPUAffinity
- Telegraf agents to ingest metrics to global metrics system
- Dropwizard configuration to handle TLS, Log rotation
- Automated deployment via SaltStack/Ansible

Hardware

- Controllers
  - 4 vCPU / 8 GiB RAM can handle even the largest clusters
- Executors
  - Primarily bare metals - upto 256 cores each
  - Bridge Networking
- Gateway
  - OSS nginx for low/moderate req/s
  - Nginx-plus for very high req/s (10k+ req/s) - frequent reloads are disruptive


Controller Host services
:443
TLS v1.3
HTTPS
NGINX for Drove console UI dynamic proxy
Nixy for Drove console UI dynamic proxy
:2181
TCP
Zookeeper
127.0.0.1:8081
HTTP
Drove metrics API (adminConnector)
:8443
TLS v1.3
HTTPS
Drove API and Console (applicationConnector)
Executor Host services
Container 1
Container 2
Docker Engine
127.0.0.1:8081
HTTP
Drove metrics API (adminConnector)
:8443
TLS v1.3
HTTPS
Drove API and Console (applicationConnector)

Maintenance

- Cluster can be put in maintenance mode
  - Can be used to bulk-upgrade software on all nodes without affecting any instances
- Executor blacklisting
  - Can be used to take executors out of rotation

Other Components

Drove-Gateway

- Gateway keeps nginx configuration updated with vhost info obtained from controllers
  - Hooks onto drove events for state updates
  - Calls drove apis on event and periodically to keep nginx cluster in sync with topology of deployed apps
  - Uses go-templating to generate routing configuration
  - Supports using runtime APIs to update upstreams to avoid reloading when using nginx-plus
- Using well-understood, scalable proxy platforms like Nginx, HAProxy

drove

Epoch

- Cron - based task scheduling for drove tasks
- Think of it being very similar to Chronos, but on Drove and not Mesos.
- Self-serve UI for creating and managing tasks
- Instant and scheduled run topologies with quartz cron expressions

https://github.com/PhonePe/epoch

drove

drove-cli

Command line interface for the Drove Container Orchestrator.

https://github.com/PhonePe/drove-cli

End to end functionality for managing:

- Drove cluster including Executor blacklisting
- Applications and application instances
- Tasks
- Event tailing
- Log tailing and downloads

drove

coredns-drove

CoreDNS plugin for Drove

https://github.com/PhonePe/coredns-drove

- Tracks leader controller
- Responds to A requests with gateway IP
- Responds to SRV records with executor host/port

How do we…

drove

Deploy

We have an approval based internal system for configuration management and deployment orchestration across multiple data centers and compliance domains

Manage auth

We have internal Oauth based system for authentication. It spans across human operators as well as systems.

Supports complicated internal and external auth models as needed by our various business, compliance and product needs.

Drove auth module is extended to use internal auth system. (Only difference with proprietary version)

drove

Store state

We don’t .. our containers are stateless

Some testing underway to use our internal document store as volume for testing environment

Handle Service Discovery

We use Ranger for service discovery from day one

Supports multiple backends like http sources, zk and drove

We have additional routing layer on top to route traffic as dictated by the DR strategies of different teams

drove

Manage Logs

Good old logrotate app logs to backup boxes

About to go live with customised version of Grafana Loki

Manage Metrics

Elaborate metrics collection and analysis system that handles 15M metrics/sec

Customised version of OpenTSDB and grafana

Anomaly detection using proprietary algorithms and models

drove

Scale Containers

Proprietary Auto Scaler that uses signals from anomaly detector and talks to the deployment system to tune number of containers across multiple Drove clusters

drove

Epilogue

Impact

- Highly stable and scalable production and testing environments
- 25-30% performance gain across Java apps
- Dimensionally better performance on ML models due to NUMA localization and core pinning
- Fast and reliable deployments

Interesting Internal Projects

- Distributed virtual device farm that spins up android emulator on the cluster using KVM inside Docker
- ML platform that manages real-time scoring ML models at huge scale
- Distributed Vulnerability assessment tool
- Distributed Locust Based Performance Testing system
- Many more…

drove

Feature Roadmap

- Overall
  - Local Services
  - Metadata and rule based routing
  - GPU resource management
- Controller
  - Remove dependence on ZK
- Drove-Gateway
  - HAProxy support
  - Multi cluster support
- CLI
  - Docker Compose like functionality
- OpenWhisk Drove backend

Questions?

drove

Thank You

For further resources visit
https://phonepe.github.io/drove-orchestrator/

drove