GIAC | BEIJING Dec.12.16-17

架構 ARCHNOTES 高可用架构

The Container Orchestration on Mesos

by Gilbert Song(宋子豪) Mesosphere

技术架构未来

thegiac.com

Outline

- Mesos overview and fundamentals

- Why should I pick Mesos?

- Containerization in Mesos
  - Unified Containerizer
  - Nested container support (aka. Pod)

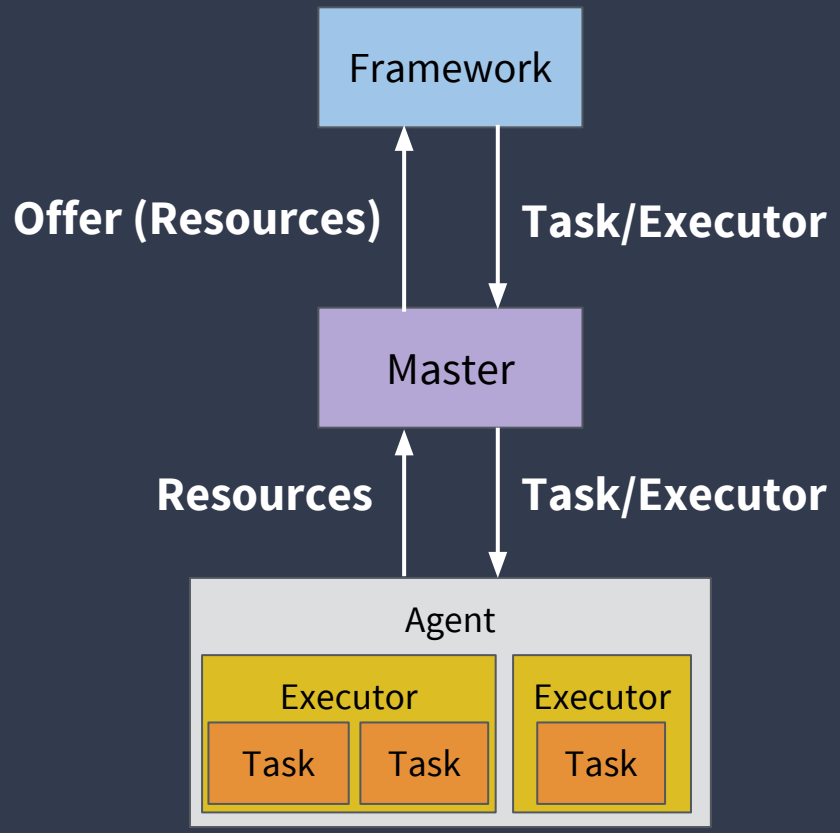
Mesos: A kernel for data center applications

- What does a traditional OS kernel provide?
  - Resource management — Host cpu, memory, etc.
  - Programming abstractions — POSIX API: processes, threads, etc.
  - Security and isolation — Virtual memory, user, etc.

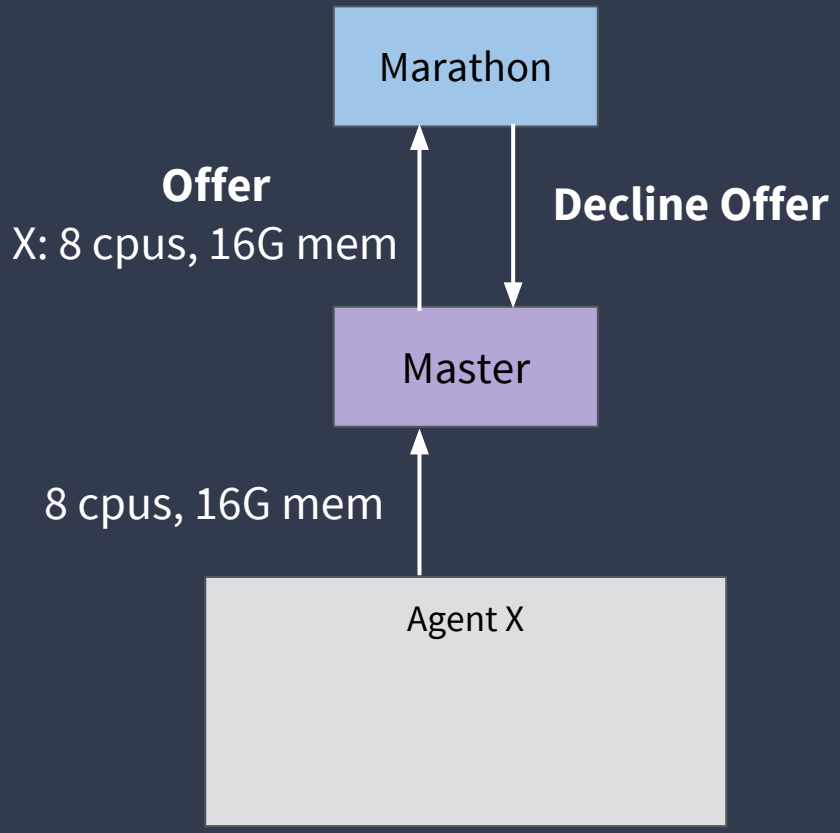
- Mesos: A kernel for data center applications
  - Resource management — Cluster cpu, memory, etc.
  - Programming abstractions — Mesos API: Task, Resource, etc.
  - Security and isolation — Containerization

Programming abstractions

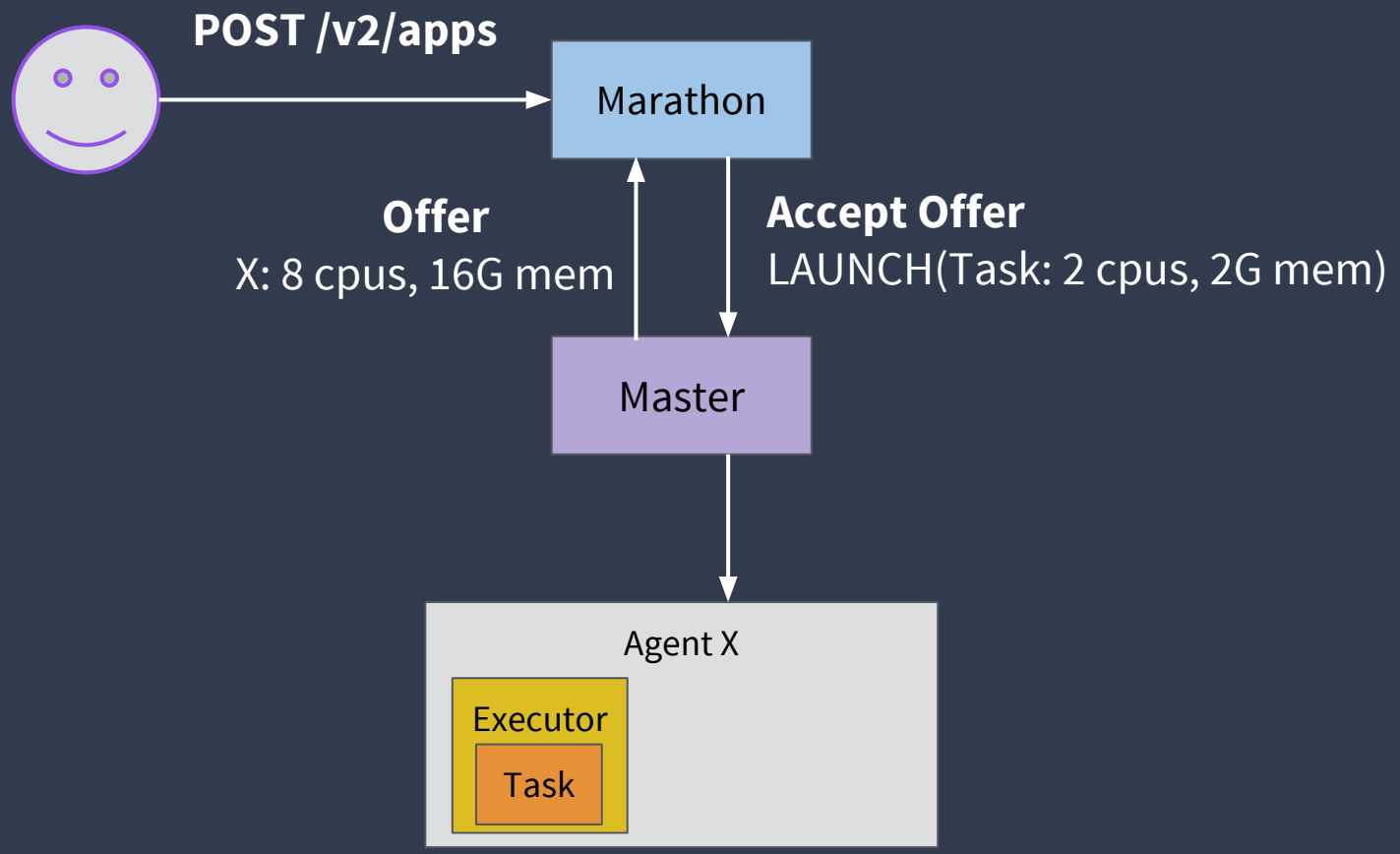
- Key concepts
  - Framework
  - Resource/Offer
  - Task
  - Executor



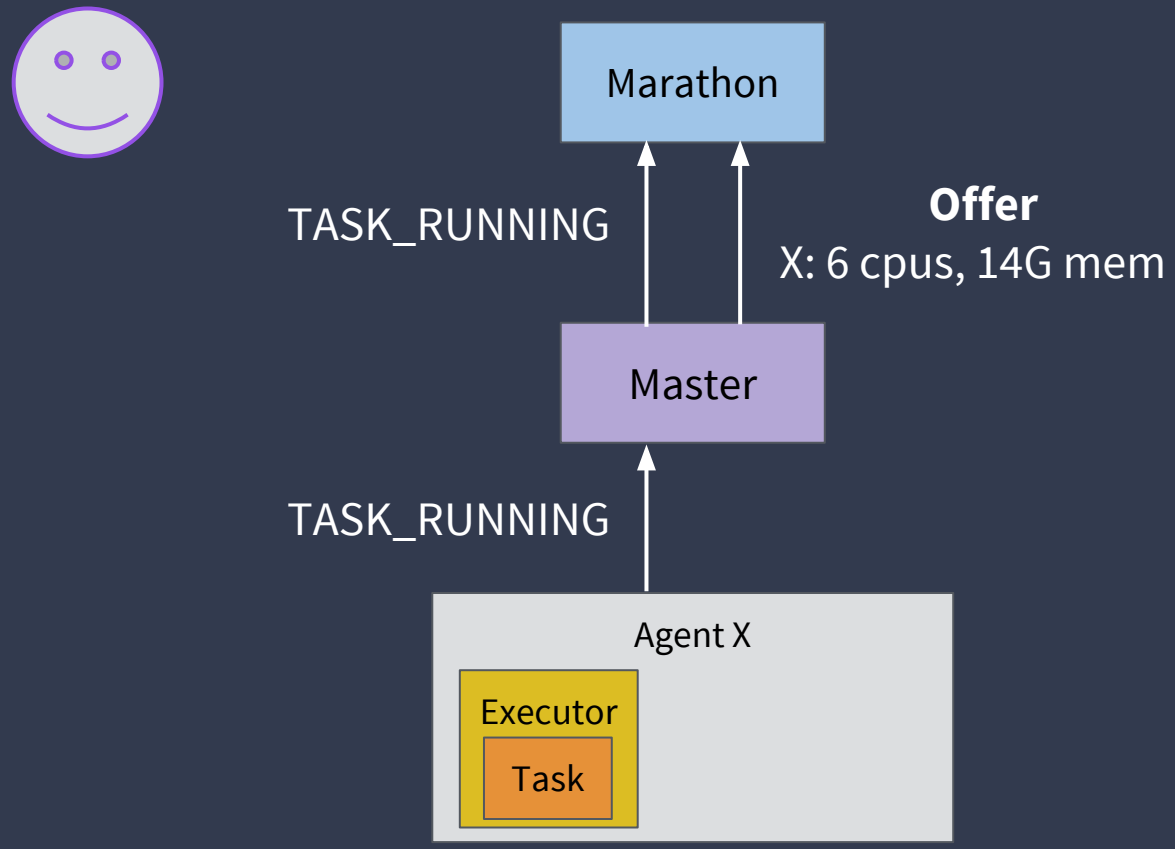
Case study: Marathon



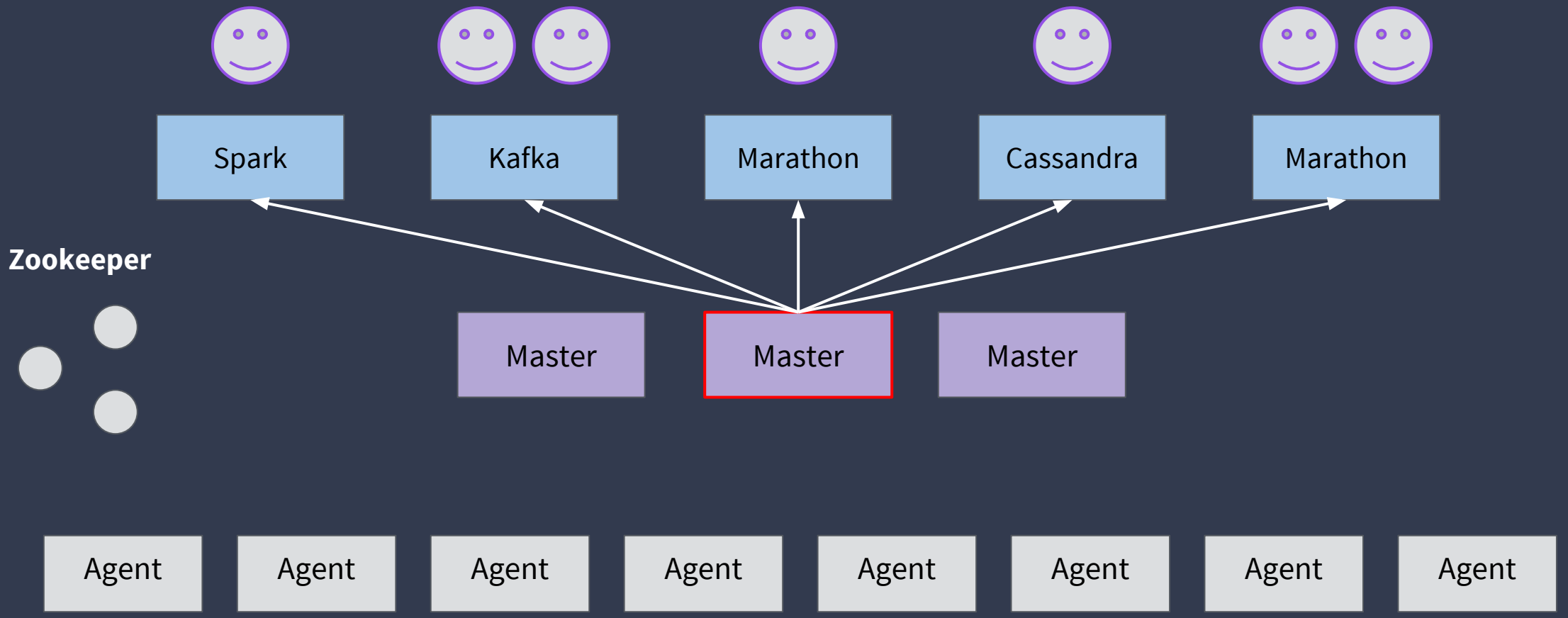
Create a Marathon app



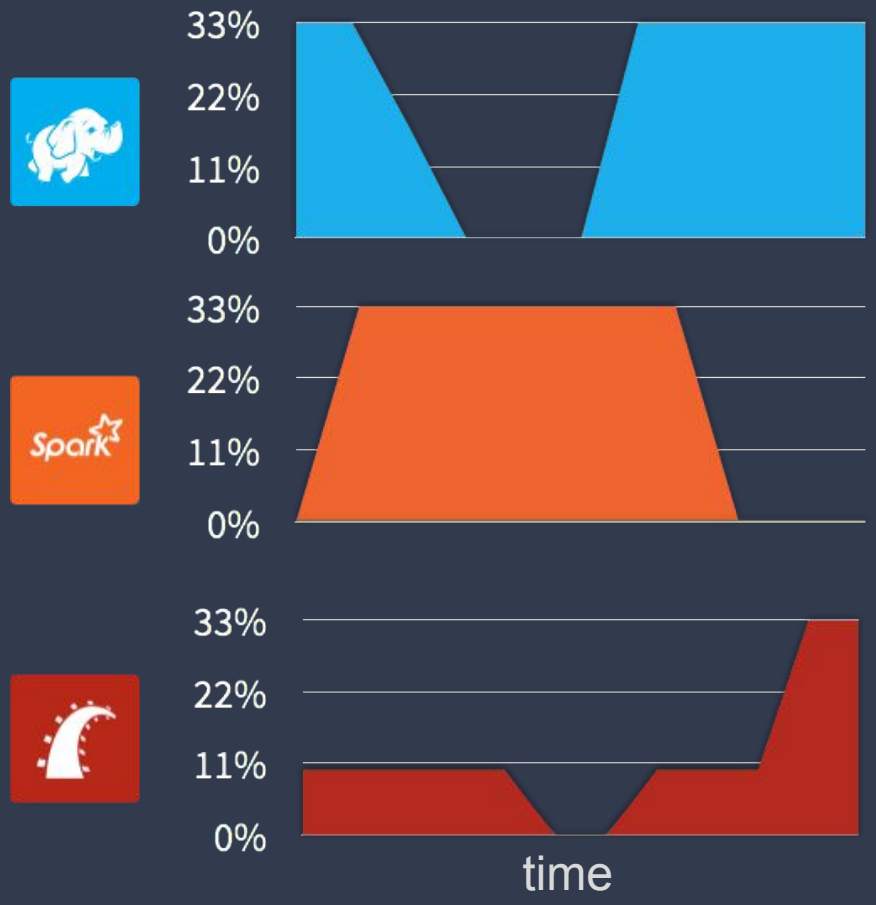
Create a Marathon app

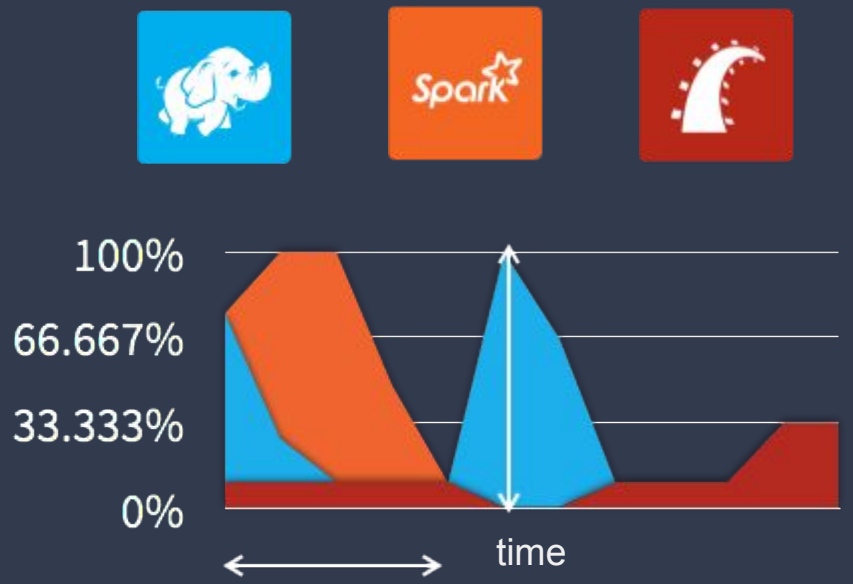


A typical Mesos cluster



Mesos helps improve cluster utilization



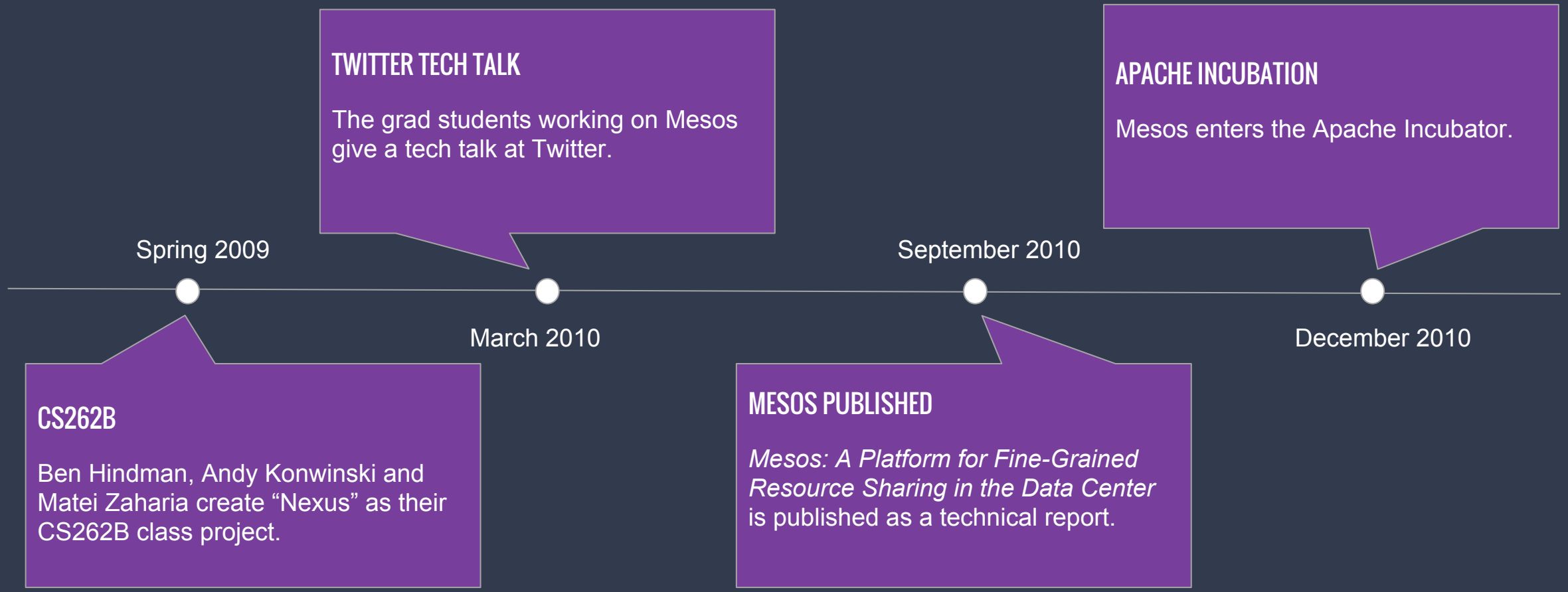


Why should I pick Mesos?

- Production ready
- Proven scalability
- Highly customizable and extensible

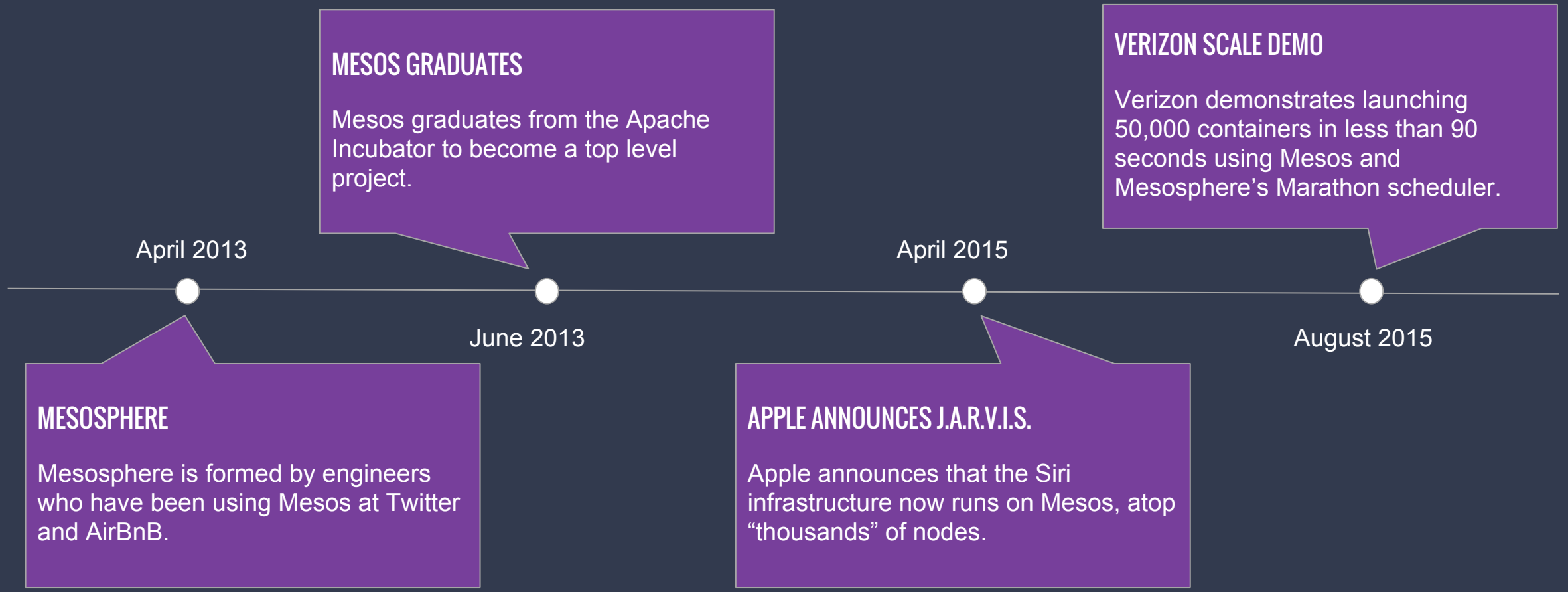
Production Ready

The history of Mesos



Why Mesos?

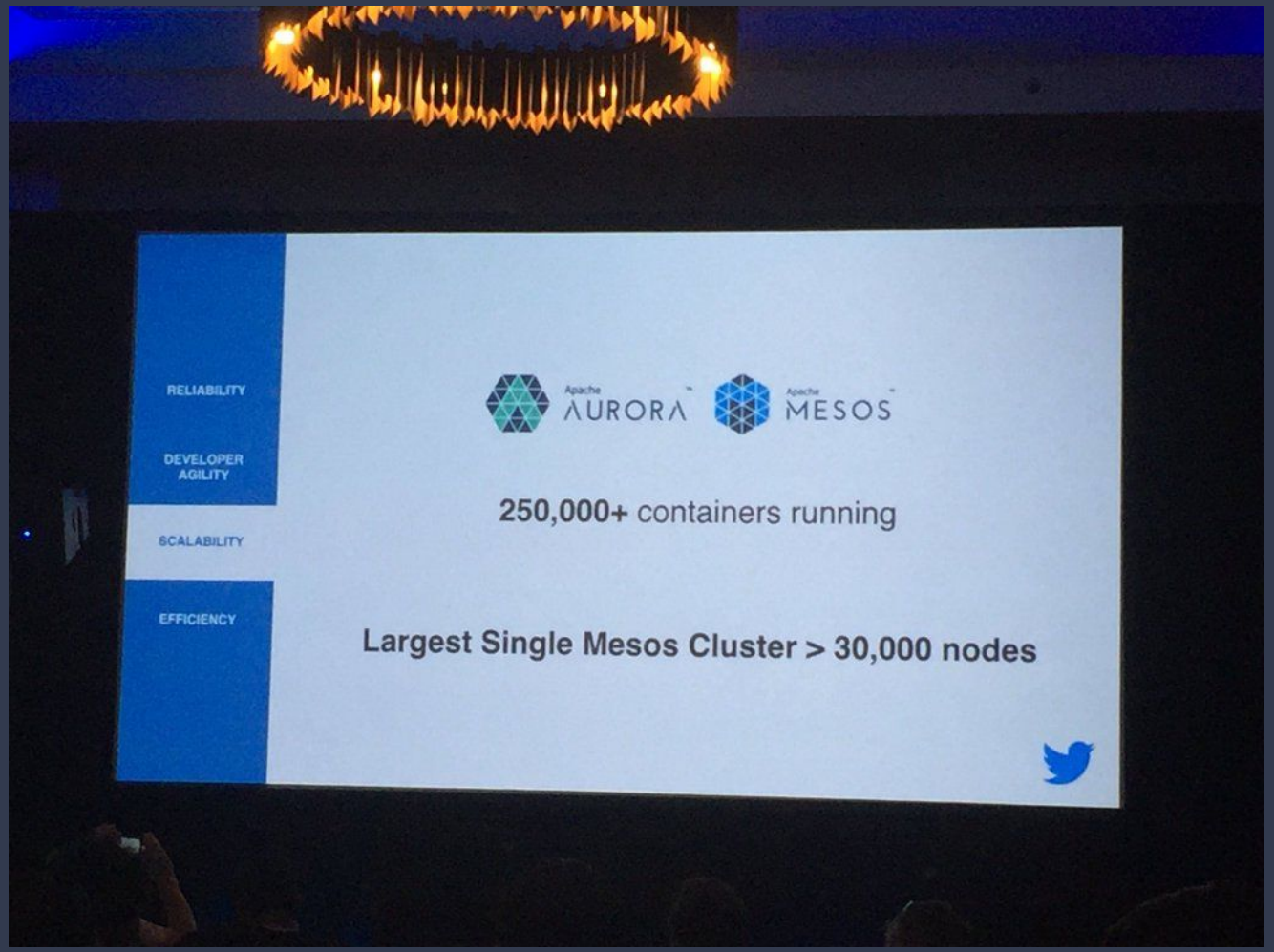
The history of Mesos



Production Mesos Users

Proven Scalability

Twitter

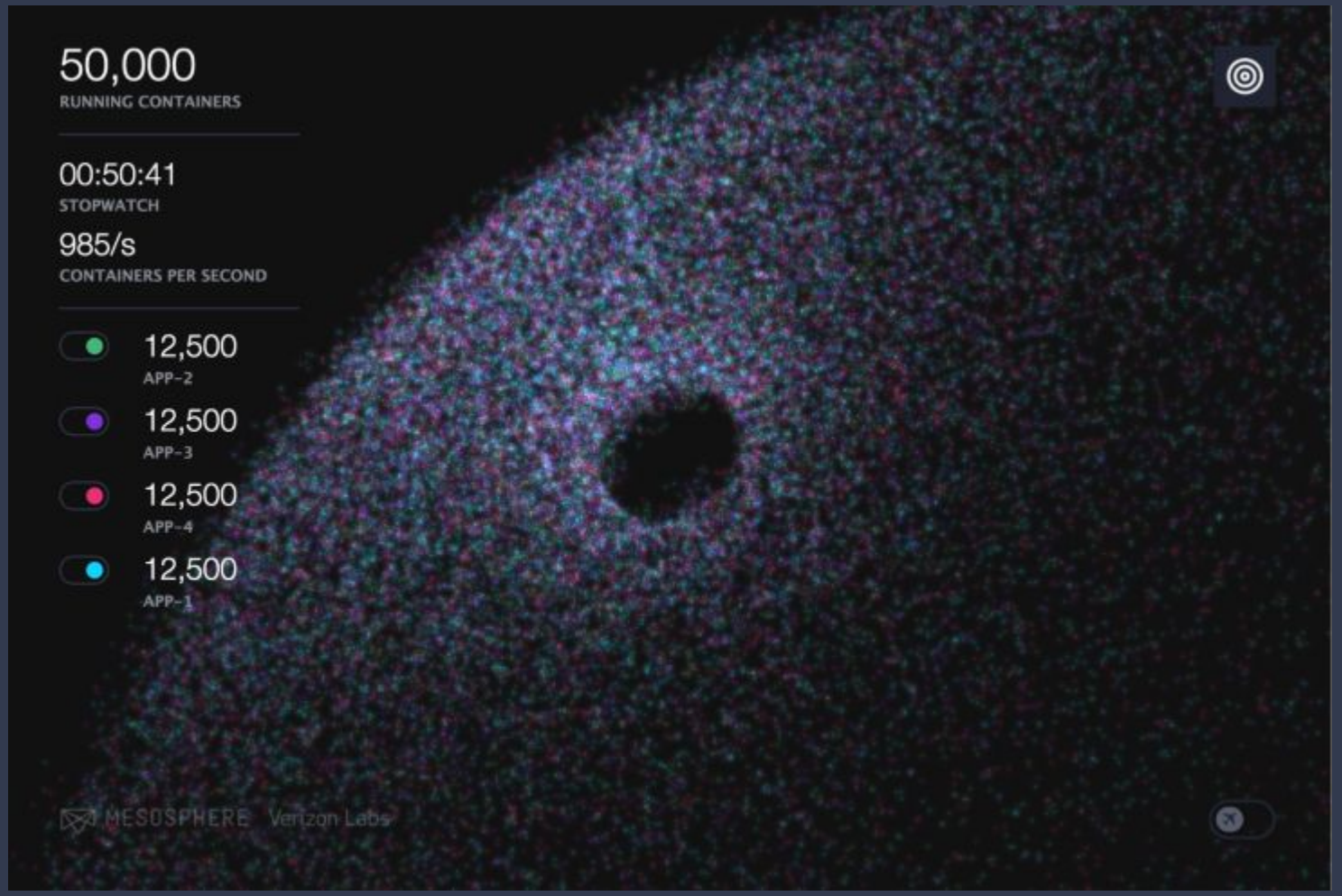


- **Largest Mesos cluster**
  - > 30000 nodes
  - > 250K containers

Apple

- **Siri is powered by Mesos!**

Verizon



- **50K containers in 50 seconds**

Why Mesos is so scalable?

- **Stateless master**
  - Inspired from the GFS design
  - Agents hold truth about running tasks (distributed)
  - Master state can be reconstructed when agents register
- **Simple, only cares about**
  - Resource allocation and isolation
  - Task management
- **Implemented in C++**
  - Native performance
  - No GC issue

What does it mean to you?

- Known that Mesos will scale to Twitter/Apple level
  - Feature is easy to add, took time to make it scalable
- Quality assurance for free
  - Imagine a test environment having 30k+ nodes with real workload
- Take backwards compatibility seriously
  - We don’t want to break their production environment

Highly Customizable and Extensible

Why this is important?

- Every company's environment is different
  - Scheduling
  - Service discovery
  - Container image format
  - Networking
  - Storage
  - Special hardware/accelerators (e.g., GPU, FPGA)

- No one-fits-all solution typically

Pluggable schedulers

- For instance, you need separate schedulers for
  - Long running stateless services
  - Cron jobs
  - Stateful services (e.g., database, DFS)
  - Batch jobs (e.g., map-reduce)

Mesos frameworks == pluggable schedulers

- Monolithic scheduler?

  Monolithic schedulers do not make it easy to add new policies and specialized implementations, and may not scale up to the cluster sizes we are planning for.

  --- From Google Omega Paper (EuroSys'13)

Flexible service discovery

- Mesos is not opinionated about service discovery
  - DNS based
  - ZK/Etcd/Chubby based (e.g., twitter, google, with client libraries)
  - Your custom way, every company is different
  - Mesos provides an endpoint to stream SD information

- DNS based solution does not scale well

  Larger jobs create worse problems, and several jobs many be running at once. The variability in our DNS load had been a serious problem for Google before Chubby was introduced.

  --- From Google Chubby paper (OSDI'06)

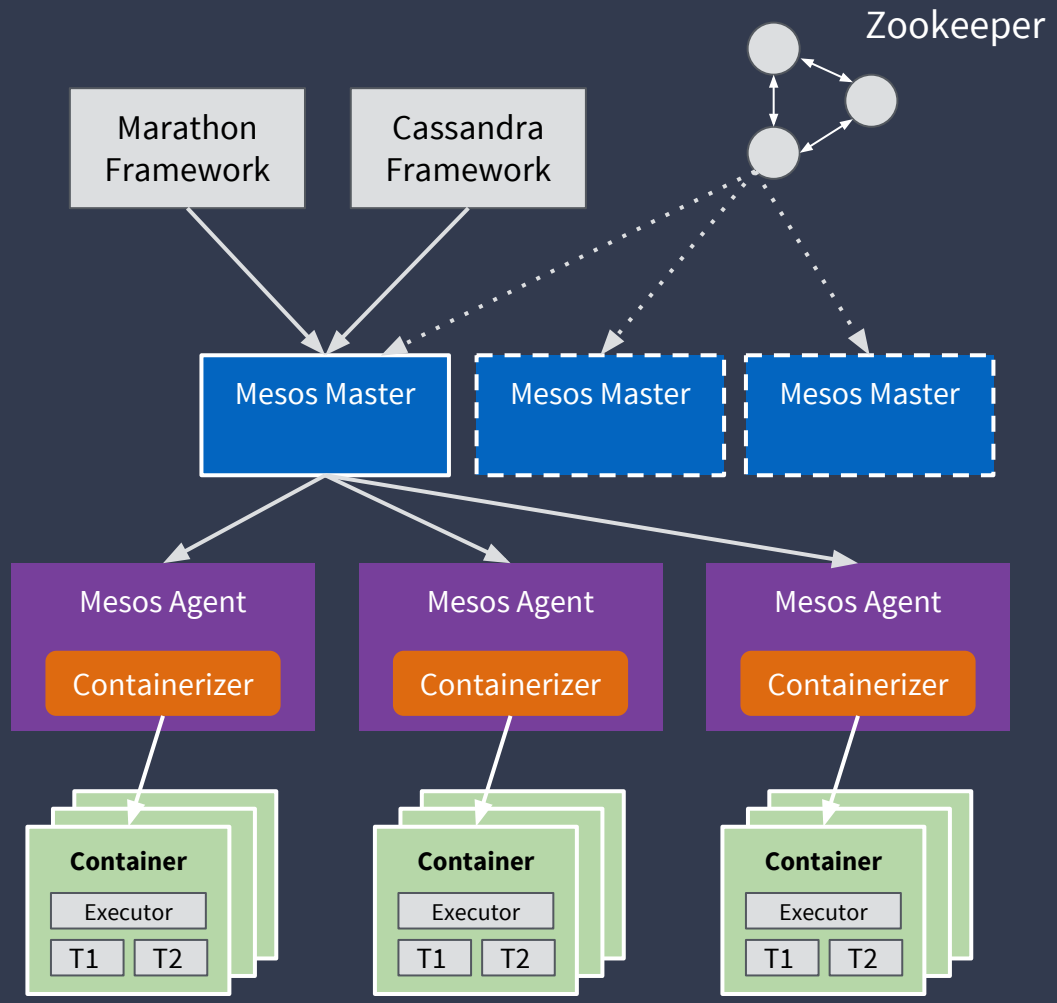
Pluggable and extensible containerization

- Container image format
- Networking
- Storage
- Security
- Custom isolation
- Container lifecycle hooks

Outline

- Mesos overview and fundamentals
- Why should I pick Mesos?
- Containerization in Mesos
  - Pluggable architecture
  - Container image
  - Container network
  - Container storage
  - Customization and extensions
  - Nesting container support (aka, Pod)

Containerizer



Containerizer

- Between agents and containers
- Launch/update/destroy containers
- Provide isolations between containers
- Report container stats and status

Currently supported containerizers

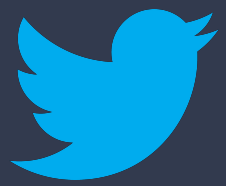
Docker containerizer

- Delegate to Docker daemon

Mesos containerizer

- Using standard OS features (e.g., cgroups, namespaces)
- Pluggable architecture allowing customization and extension

Very stable. Used in large scale production clusters



Currently supported containerizers

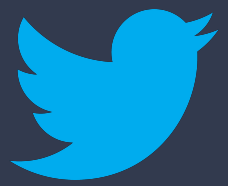
Docker containerizer

- Delegate to Docker daemon

Mesos containerizer

- Using standard OS features (e.g., cgroups, namespaces)
- Pluggable architecture allowing customization and extension
- Support Docker, Appc, OCI (soon) images natively w/o dependency

Very stable. Used in large scale production clusters



Currently supported containerizers

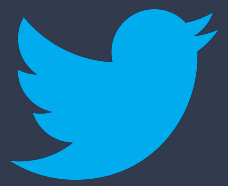
Docker containerizer

- Delegate to Docker daemon

Unified containerizer

- Using standard OS features (e.g., cgroups, namespaces)
- Pluggable architecture allowing customization and extension
- Support Docker, Appc, OCI (soon) images natively w/o dependency

Very stable. Used in large scale production clusters



Why Unified Containerizer

- Maintaining two containerizers is hard
- Docker daemon getting more complex
- Support other container image format
- Allow customization and extension

Maintaining multiple containerizers is hard

- New features require changes to both containerizers
  - E.g., GPU support, Persistent volumes, etc.
  - Code duplication → bugs, maintenance issue

- Coordination needed between two containerizers
  - Global resources like GPU, net_cls handles, etc.

Docker daemon getting more complex → stability issue

Hacker News new | threads | comments | show | ask | jobs | submit

▲ What I found wrong in Docker 1.12 (linux-toys.com)

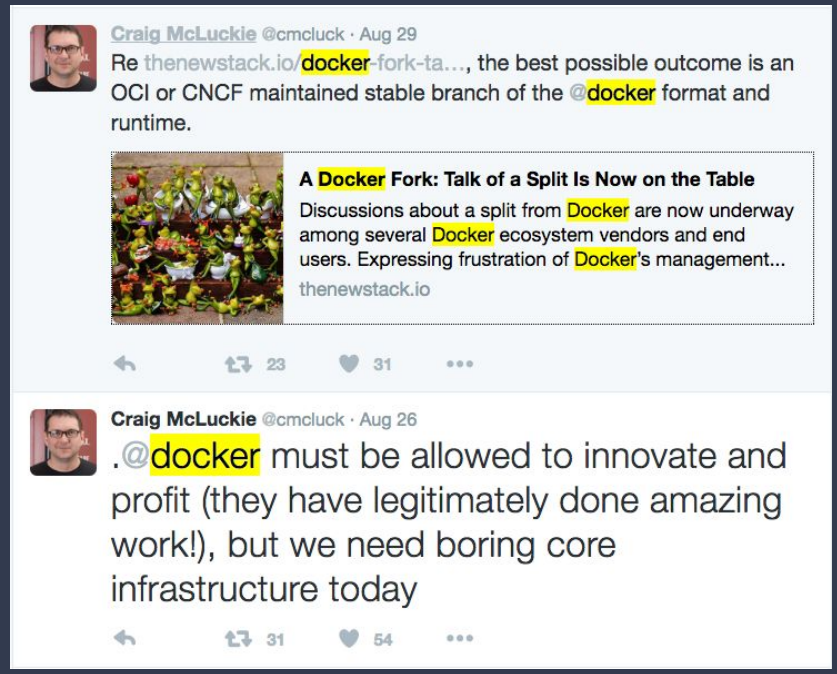
390 points by rusher81572 17 days ago | hide | past | web | 228 comments | favorite

▲ andrewguenther 17 days ago [-]

Disclaimer: I work at AWS, but on a product which does not compete with Docker or

I wouldn't even limit this to just the swarm feature. We've been running Docker in p release. We had to upgrade directly from Docker 1.7 to 1.11 because every release backporting features which were worth the risk.

Speaking of 1.12, my heart sank when I saw the announcement. Native swarm adds shove these new tools down everyone's throats and really made it feel like they saw containers." I get the feeling we'll be running 1.11 for quite some time...



Support other container image format

- Docker only supports Docker image format

- Other image formats we want to support
  - OCI image spec
  - Appc image spec
  - CVMFS
  - Host filesystem with tars/jars
  - Your own image format!

Allow customization and extension

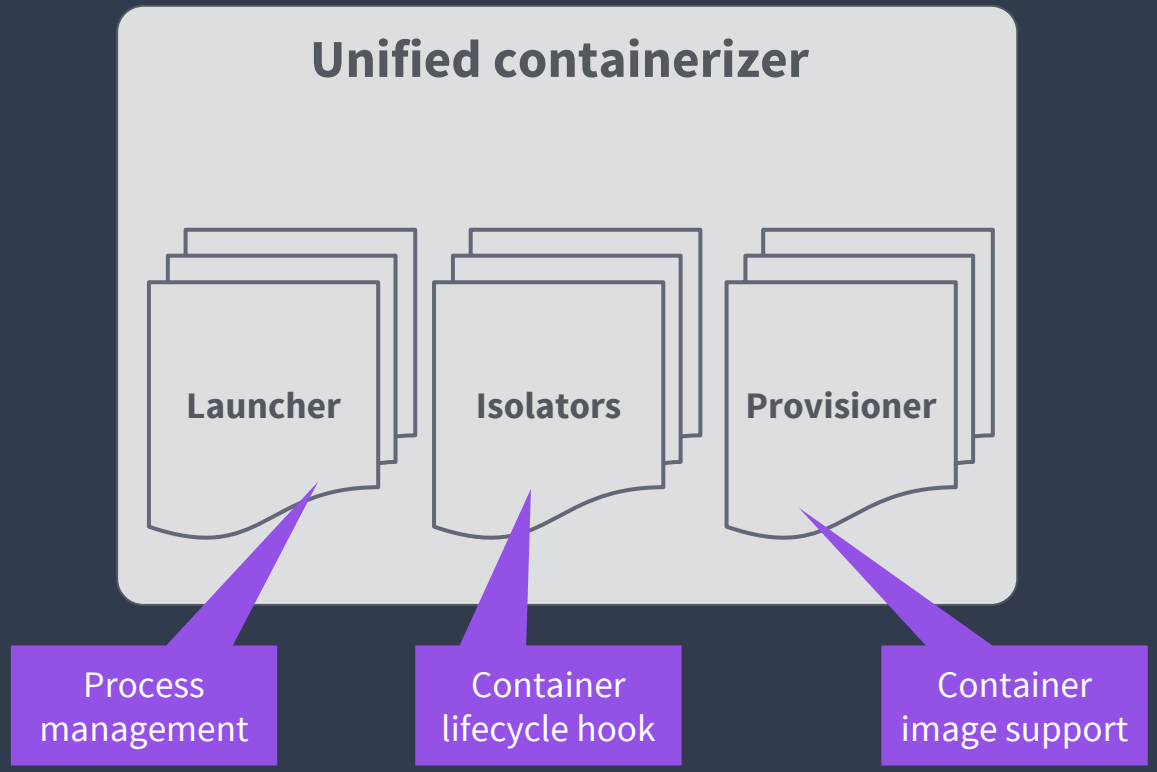
- Docker only allows certain components to be extended
  - Network (Libnetwork)
  - Storage (DVD)

- Mesos needs more than that
  - Disk quota enforcement
  - Security extensions
  - Special hardware support (e.g., GPU)
  - ...

Unified Containerizer

- Pluggable architecture
- Container image
- Container network
- Container storage
- Customization and extensions
- Nesting container support

Pluggable architecture



Launcher

Responsible for process management

- Spawn containers
- Kill and wait containers

Supported launchers:

- Posix launcher
- Linux launcher
- Windows launcher

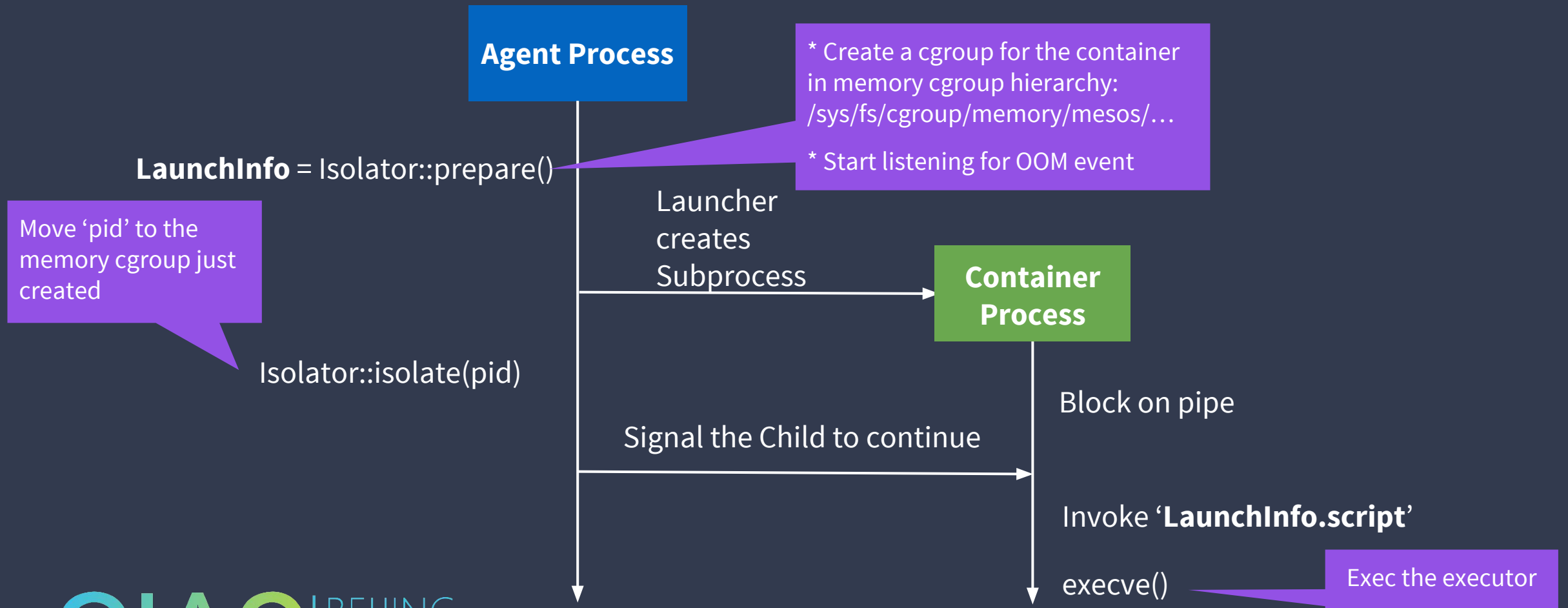
Isolator

Interface for extensions during the **life cycle** of a container

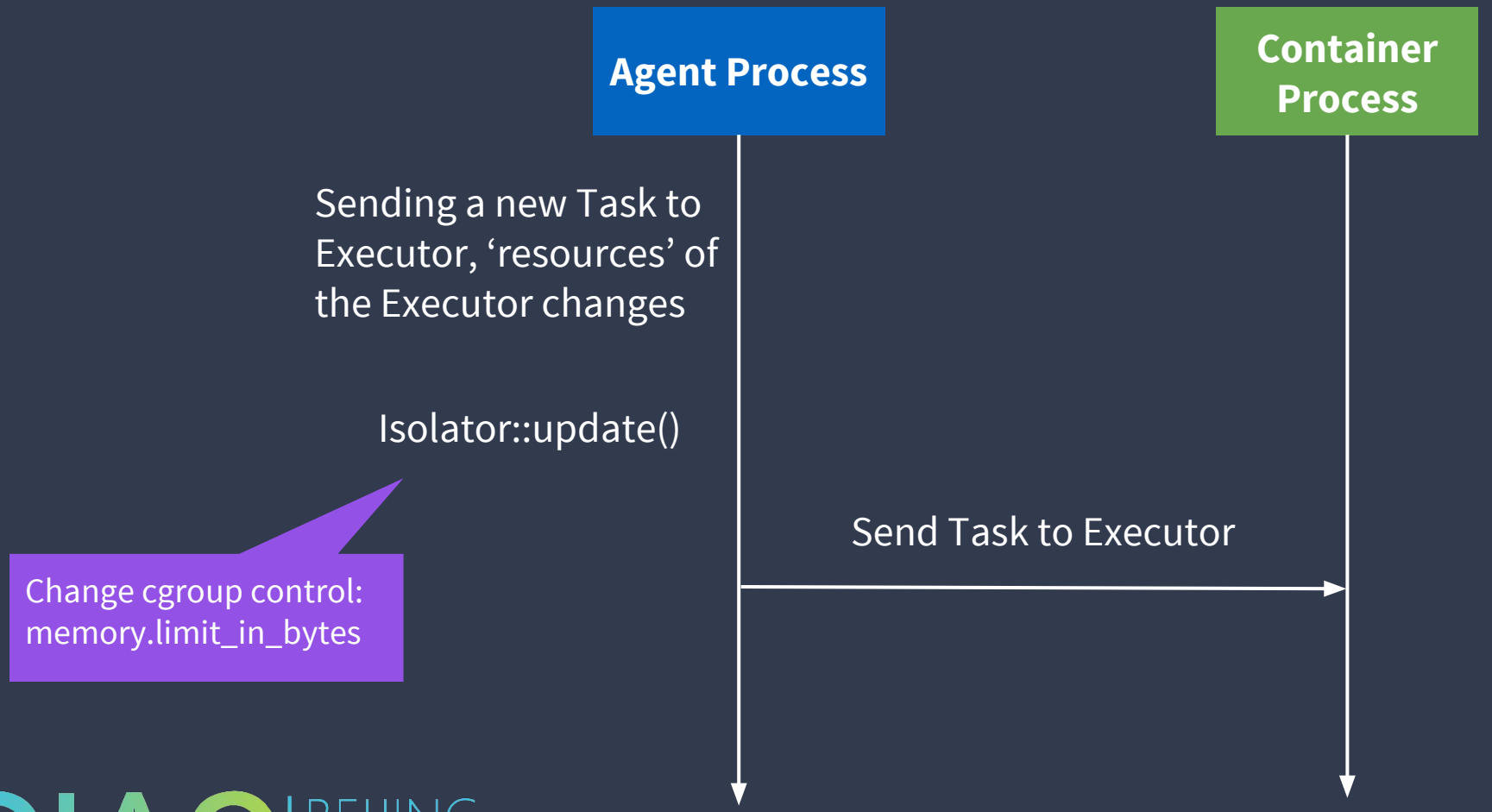
- Pre-launch - prepare()
- Post-launch (both in parent and child context) - isolate()
- Termination - cleanup()
- Resources update - update()
- Resources limitation reached - watch()
- Agent restart and recovery - recover()
- Stats and status pulling - usage()

Sufficient for most of the extensions!

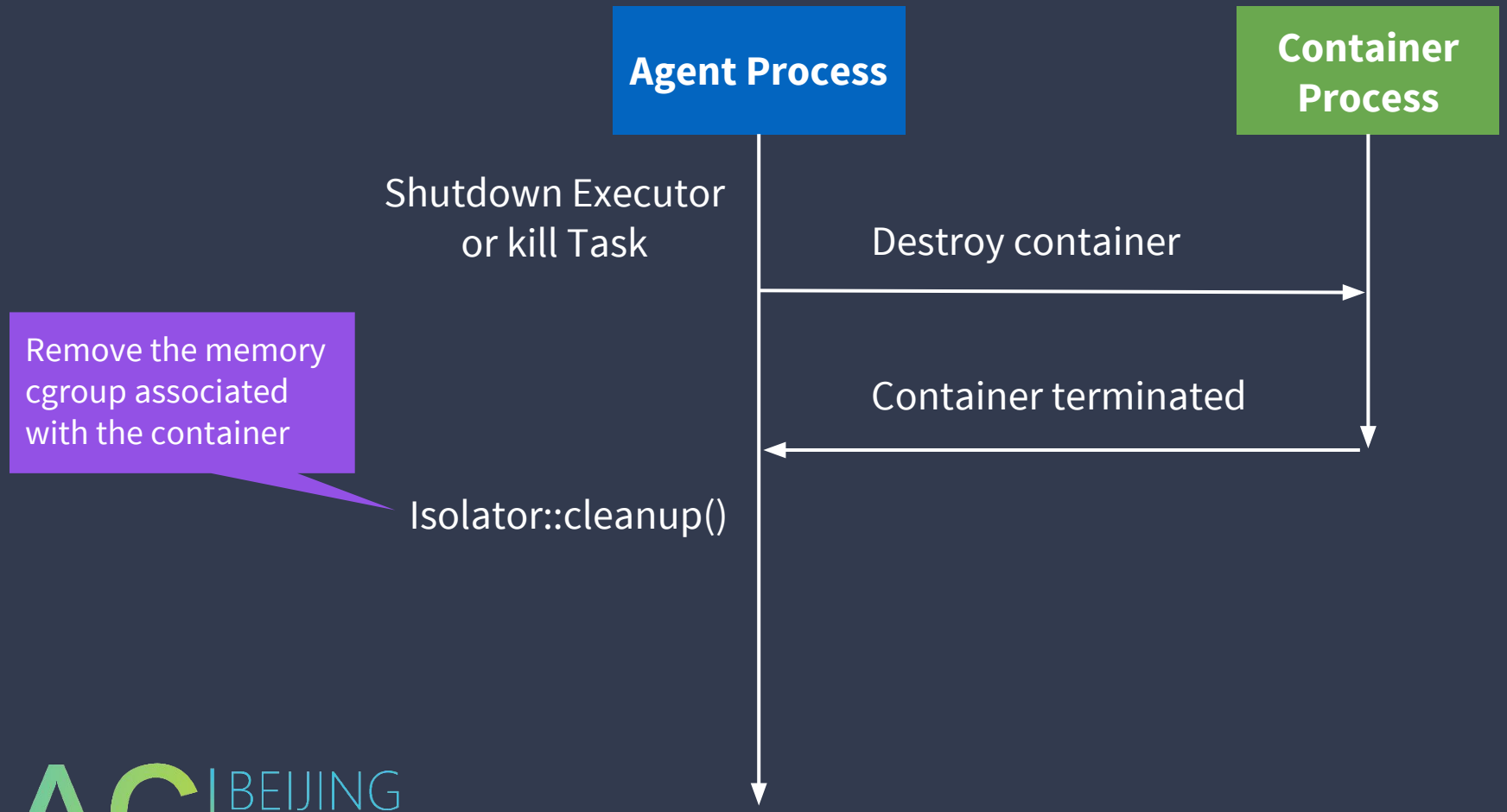
Isolator example: cgroups memory isolator



Isolator example: cgroups memory isolator



Isolator example: cgroups memory isolator



Built-in isolators

Cgroups isolators:	`cgroups/cpu, cgroups/mem, ...`
Disk isolators:	`disk/du, disk/xfs`
Filesystem isolators:	`filesystem/posix, filesystem/linux`
Volume isolators:	`docker/volume, ...`
Network isolators:	`network/cni, network/port_mapping`
GPU isolators:	`gpu/nvidia`
Namespace isolators:	`namespaces/pid, namespaces/cgroup, ...`
Misc isolators:	`linux/capabilities, linux/rlimits`

..... and more! Need your contribution!

Container image support

Start from 0.28, you can run your Docker container on Mesos without a Docker daemon installed!

- One less dependency in your stack
- Agent restart handled gracefully, task not affected
- Compose well with all existing isolators
- Easier to add extensions

Pluggable container image format

- Mesos supports multiple container image format
  - Docker (without docker daemon)
  - Appc (without rkt)
  - OCI (ready soon)
  - CVMFS (experimental)
  - Host filesystem with tars/jars
  - Your own image format!

Used in large scale production clusters

Provisioner

- Manage container images
  - Store: fetch and cache image layers
  - Backend: assemble rootfs from image layers
    - E.g., copy, overlayfs, bind, aufs
- Store can be extended
  - Currently supported: Docker, Appc
  - Plan to support: OCI (ongoing), CVMFS
  - Custom fetching (e.g., p2p)

Container network support

- Support Container Network Interface (CNI) from 1.0
  - A spec for container networking
  - Supported by most network vendors

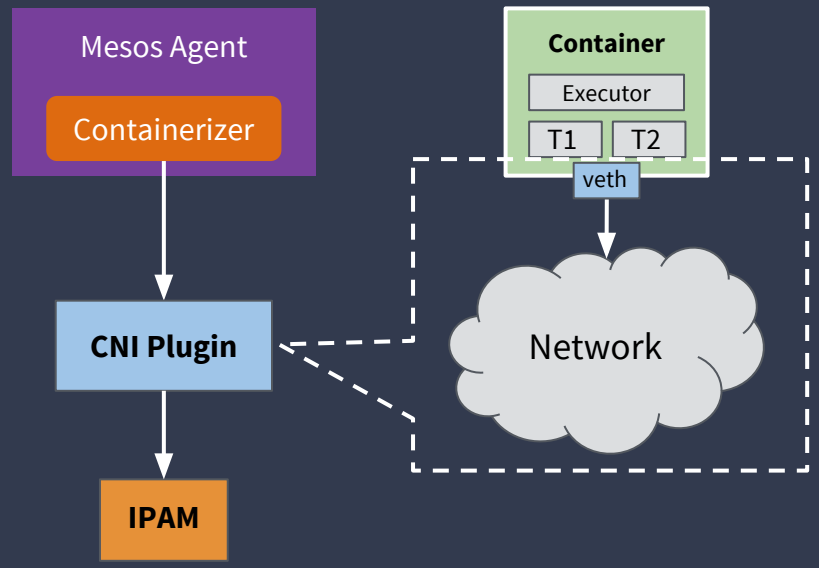
- Implemented as an isolator
  - `--isolation=network/cni,...`

Container Network Interface (CNI)

- Proposed by CoreOS :

  https://github.com/containernetworking/cni

- Simple contract between container runtime and CNI plugin defined in the form of a JSON schema
  - CLI interface
  - ADD: attach to network
  - DEL: detach from network



Why CNI?

- Simpler and less dependencies than Docker CNM
- Backed by Kubernetes community as well
- Rich plugins from network vendors
- Clear separation between container and network management
- IPAM has its own pluggable interface

CNI plugins

Existing CNI plugins

- ipvlan
- macvlan
- bridge
- flannel
- calico
- contiv
- contrail
- weave
- …

You can write your own plugin, and Mesos supports it!

Container storage support

- Support Docker volume plugins from 1.0
  - Define the interface between container runtime and storage provider
  - https://docs.docker.com/engine/extend/plugins_volume/
- A variety of Docker volume plugins
  - Ceph
  - Convoy
  - Flocker
  - Glusterfs
  - Rexray

Container security support

- Mesos 1.1 supports Linux capabilities
  - Allow the container to run as root with limited capabilities
- Plan to support seccomp and user namespaces!
- Join the other talk!

Extensions

Launcher

- Custom container processes management

Isolator

- Extension to the life cycle of a container

Provisioner

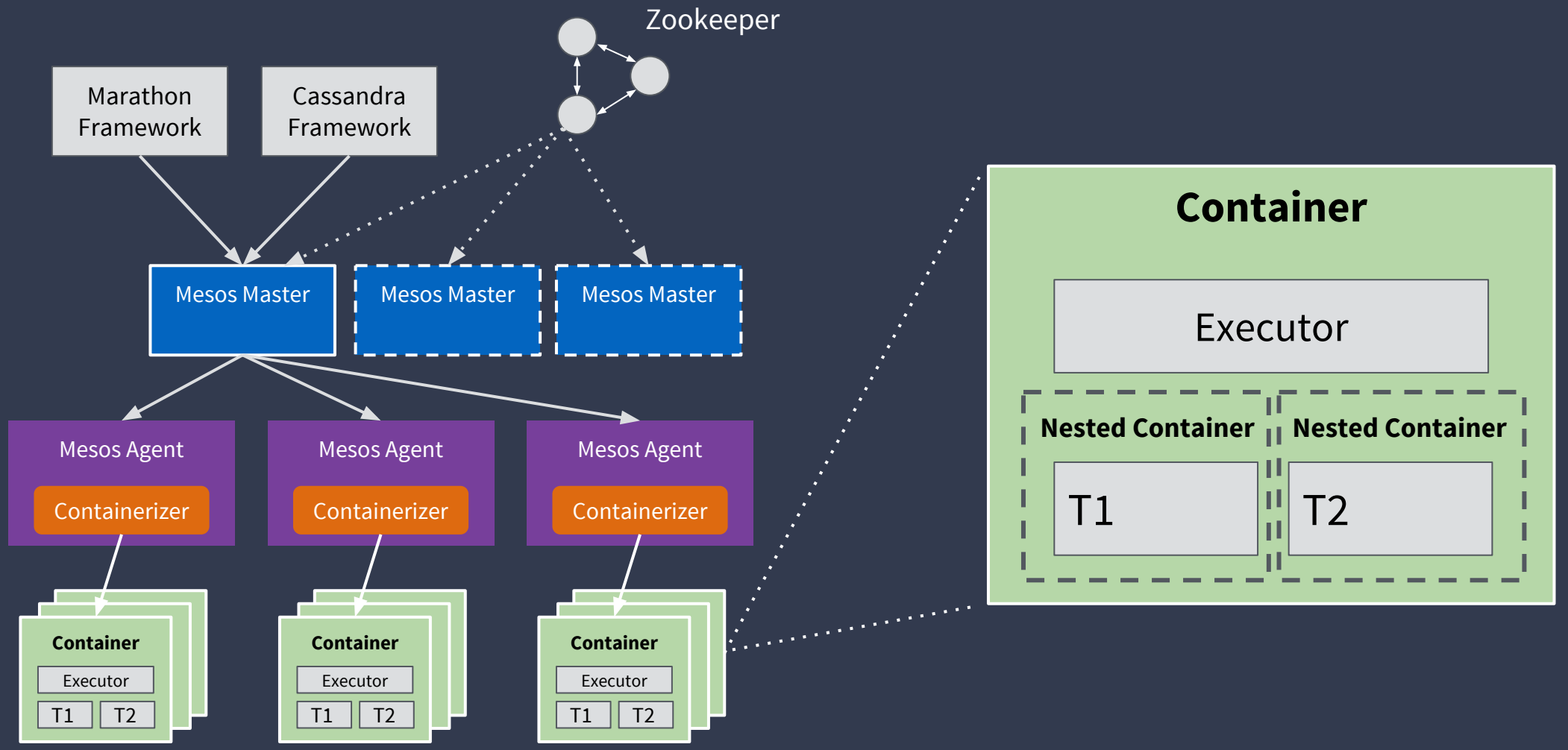
- New type of images
- Custom fetching and caching

Nested container support (aka, Pod)

- New in Mesos 1.1
  - Building block for supporting Pod like feature

- Highlighted features
  - Support arbitrary levels of nesting
  - Re-use all existing isolators
  - Allow dynamically creation of nested containers

Nested container support



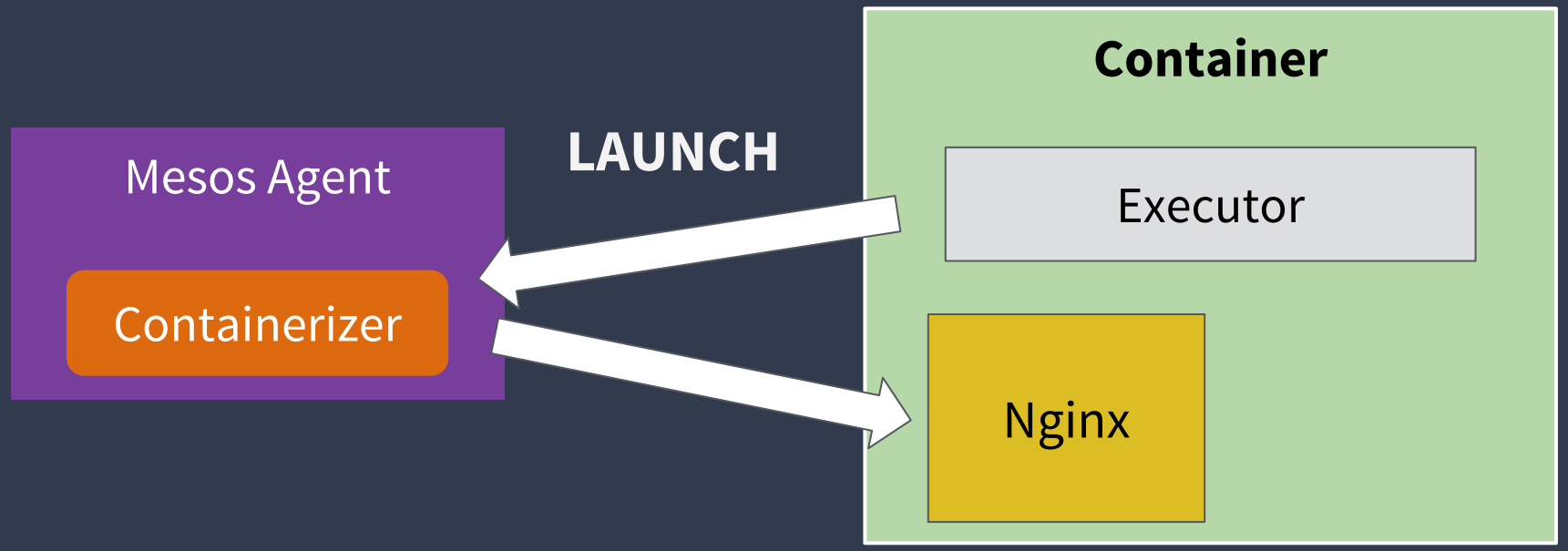
Nested container support

New Agent API for Nested Containers

```
message agent::Call {
  enum Type {
    // Calls for managing nested containers
    // under an executor's container.
    LAUNCH_NESTED_CONTAINER = 14;
    WAIT_NESTED_CONTAINER = 15;
    KILL_NESTED_CONTAINER = 16;
  }
}
```

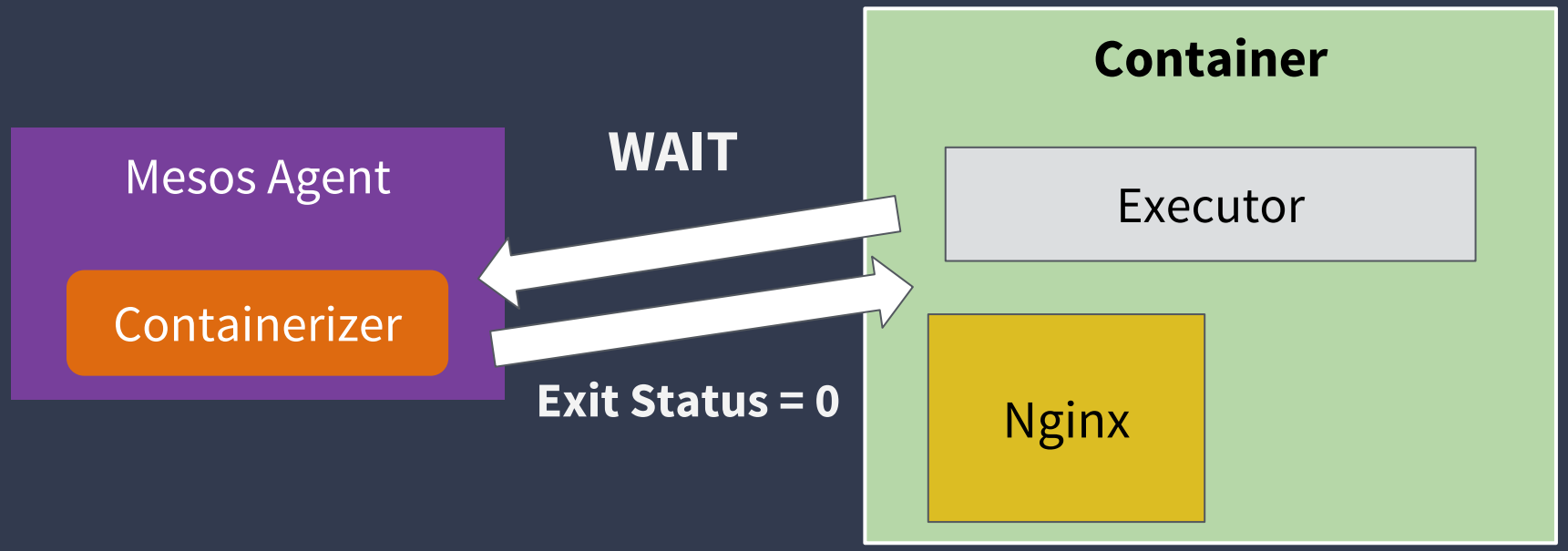
Nested container support

Launch nested container



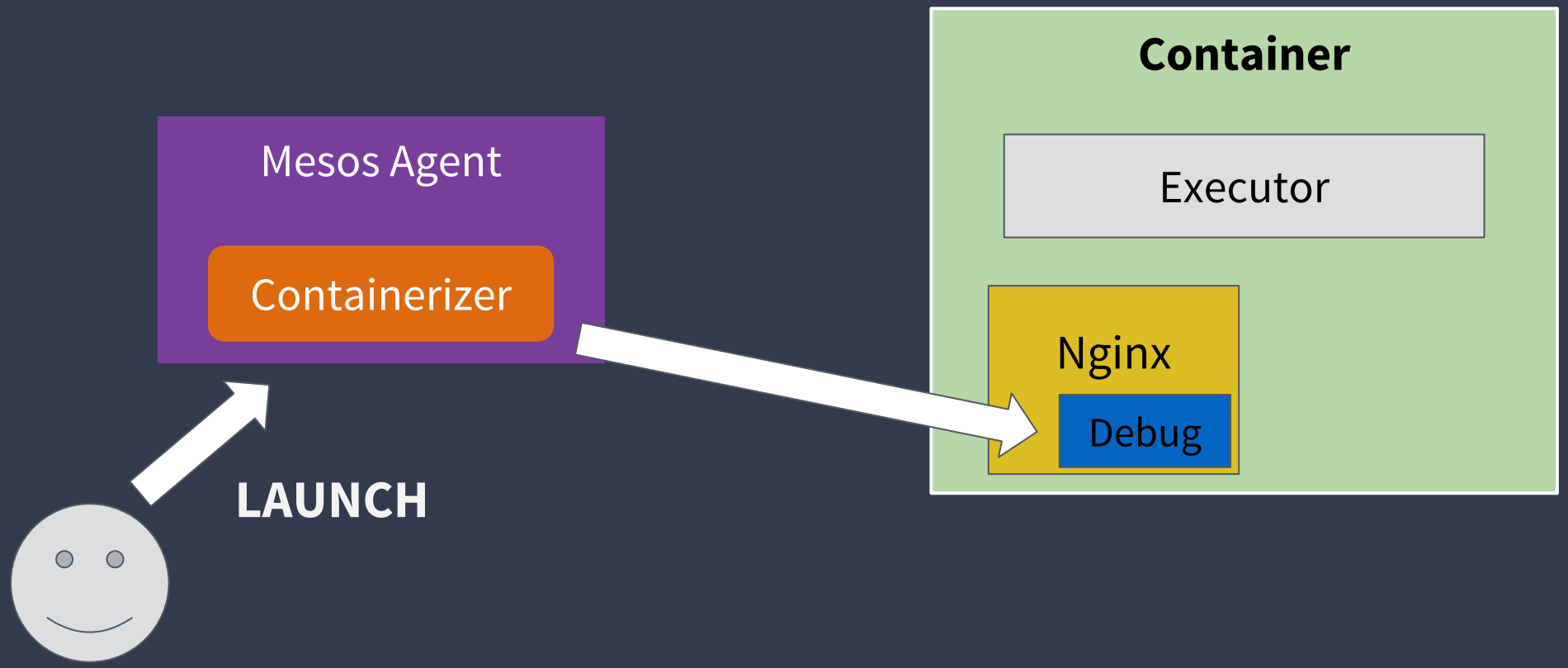
Nested container support

Watch nested container



Nested container support

Arbitrary levels of nesting



Demo

Questions?

GIAC BEIJING Dec.12.16-17

技术架构未来

架構 ARCHNOTES 高可用架构

Thank you!

songzihao888358

Email: gilbert@mesosphere.io