Container Orchestration

AGENDA

- CONTAINERIZATION ALL THE THINGS
- WHAT IS CONTAINER ORCHESTRATION?
- HOW DOES IT WORK?
- CHOOSING THE RIGHT TOOL
- HANDS ON SESSION

CONTAINERIZATION ALL THE THINGS

If we want to be able to deploy and scale easily and efficiently, we want our services to be small. Smaller things are easier to work with. Today, we are moving towards smaller, easier to manage, and shorter lived services.

But, the more things to deploy, more problems for the infrastructure team to configure and monitor.

Because **containers are meant to be short-lived**, our applications are built with scaling and fault tolerance in mind, so a temporary loss of a single containers is not a problem.

Solutions like Docker, Podman, Cri-O, and Buildah provide great flexibility to containerize and ship applications.

CONTAINERIZATION ALL THE THINGS

Containerized applications **should be able to scale up and down** based on application resource requirements, and **each service is self-sufficient** does not create infrastructure chaos.

With microservices packed inside containers and deployed to a cluster, there is a need for a different set of tools. **There is the need for cluster orchestration.**

An automated way to deploy and scale **services that will get rescheduled in case of a failure**. If a container stops working, it will be deployed again. If a whole node fails, everything running on it will be moved to a healthy one.

WHAT IS CONTAINER ORCHESTRATION?

Container orchestration is all about managing the life cycles of containers, especially in large, dynamic environments. Teams use container orchestration to control and automate many tasks like:

- Provisioning, deployment, redundancy, availability of containers
- Scaling up or removing containers to spread application load evenly across host infrastructure
- Move of containers between hosts if there is a shortage of resources in a host, or if a host dies
- Allocation of resources between containers
- Load balancing of service discovery between containers
- Health monitoring of containers and hosts

WHAT IS CONTAINER ORCHESTRATION?

The beauty of container orchestration tools is that you can use them in any environment in which you can run containers. And containers are supported in just about any kind of environment these days, from traditional on-premise servers to public cloud instances running in Amazon Web Services (AWS), Google Cloud Platform (GCP), or Microsoft Azure.

Once the container is running on the host, the orchestration tool manages its lifecycle according to the specifications you laid out in the container's definition file (for example, its Dockerfile).

HOW DOES IT WORK?

When you use a container orchestration tool, like Kubernetes or Docker Swarm (more on these shortly), you typically describe the configuration of your application in a YAML or JSON file, depending on the orchestration tool.

These configurations files are where you tell the orchestration tool where to gather container images (for example, from Docker Hub), how to establish networking between containers, how to mount storage volumes, and where to store logs for that container, how to establish networking between containers, how to mount storage volumes, and where to store logs for that container.

HOW DOES IT WORK?

Typically, teams will branch and version control these configuration files so they can deploy the same applications across different development and testing environments before deploying them to production clusters.

Containers are deployed onto hosts, usually in replicated groups. When it’s time to deploy a new container into a cluster, the orchestration tool schedules the deployment and looks for the most appropriate host to place the container based on predefined constraints (CPU or memory availability). You can even place containers according to labels or metadata, or according to their proximity in relation to other hosts—all kinds of constraints can be used.

CHOOSING THE RIGHT TOOL

- ❏ Networking
- ❏ High availability
- ❏ Ease of deployment & maintenance
- ❏ Scalability
- ❏ Service discovery
- ❏ Security & Compliance
- ❏ Support (Community & Enterprise)
- ❏ Administrative overhead

CHOOSING THE RIGHT TOOL

Amazon ECS
FROM AMAZON

Azure Container Services
FROM MICROSOFT

Docker Swarm
DOCKER OPENSOURCE TOOLS

Google Container Engine
FROM GOOGLE CLOUD PLATFORM

Kubernetes
DOCKER OPENSOURCE TOOLS

CoreOS Fleet
FROM COREOS

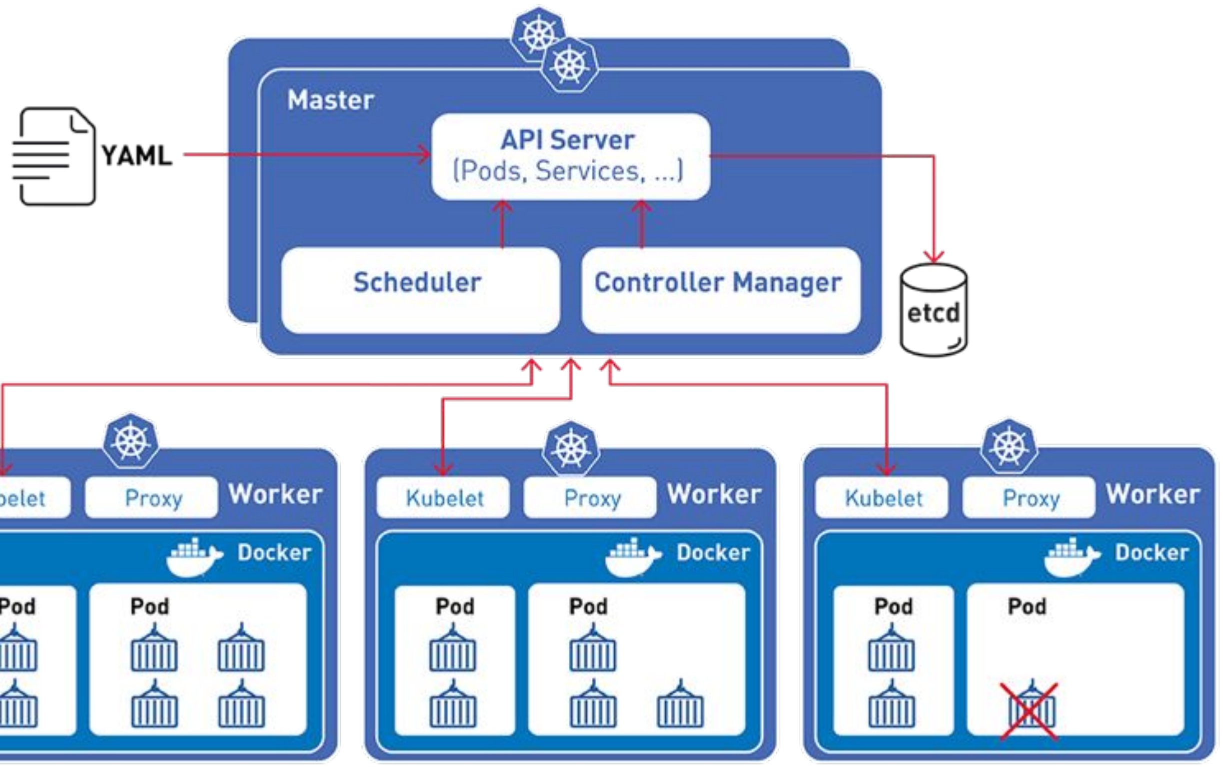
Mesosphere Marathon
FROM MARATHON

Cloud Foundry's Diego
FROM CLOUD FOUNDRY

WHAT KUBERNETES IS?

Open-Source **container orchestration** that provides application deployment, scaling, and management.

Designed from the ground up to be an environment for building distributed applications from containers.



https://kubernetes.io/

HOW TO USE KUBERNETES?

Minikube a tool that makes it easy to run K8s locally. It runs a single-node K8s cluster inside a VM.

```
$ minikube start
```

```
$ minikube dashboard
```

Kubectl is a CLI command line interface for running commands against Kubernetes clusters.

```
$ kubectl cluster-info
```

```
$ kubectl get nodes
```

```
$ kubectl get all
```

KUBERNETES BASIC COMMANDS

Now we are going to review some basic k8s command for deployment and management.

```
$ kubectl run ghost --image=ghost:0.9
```

```
$ kubectl expose deployment ghost --port=2368 --type=LoadBalancer
```

Because we are using **Minikube** use --external-ip=$(minikube ip) flag when exposing

```
$ kubectl scale --replicas=3 deployment ghost
```

Here some pod specific access and management commands

```
$ kubectl label pods <pod-name> owner=tester
```

```
$ kubectl logs <pod-name>
```

```
$ kubectl exec -ti <pod-name> /bin/bash
```

endava | be more

HANDS ON SESSION

kubernetes

https://github.com/twogg-git/k8s-selenium

Thanks!