PEEP

------

The Cold War for Resources in Docker.

I want to be a Docker Container. Because then I’ll get all the things I want–but of course, only under the circumstance that we’re only a few containers. When we should become many, poor us, no way I want to be one anymore. There is no chance they can do a great job on a crowded host. It’s because Docker Containers get all the same amount of resources–"NOTHING"–sharing is carrying. Even the thickest containers won’t get more resources than the thinnest. It’s like being an elephant that gets a bottle of water a day, so will ants and all the other animals on planet earth. What happens? Elephants will die while ants will waste most of their water and build swimming pools of the size of an elephant enjoying life. The End: tension at first, then, containers die, one by one, finally their host. Life ends, for all, but there’s an easy way out that I didn’t pay attention to when I first started dockerizing every service, every app and every admin tool. That is, leading my container-army.

# Leading and Managing Docker Containers.

It became a nightmare. Whenever our release manager started working, our services were getting slow, Jenkins deployment jobs turned red and some docker daemon started to hang. The reason was simply the lack of resources all containers had to share. “Bianca”, I said. She is the release manager. “I know, it might sound like a silly question, but how many feature branches at the same time do you need up and running?”. “I don’t know”, she replied. Well, not exactly the answer I hoped for but she was right. How could she know the answer to my question. It’s my responsibility to set limits depending on how much resources are available.

# Resource Management in Docker

## CPU

You run one Docker Container and set no CPU limits for it means it’ll get 100% CPU whenever it needs it.   
You run two Docker Containers and set no CPU limits for them means both will get 50% CPU when they both need it at the same time.  
You run two Docker Containers and set no CPU limits for them but only one of them is working means the working Docker Container gets 100%, the other 0% CPU because it doesn’t need any.

You run three Docker Containers and actually set CPU limits. One gets 1024 shares, which is the relative default value and two get 512 shares. Are all three working will mean that one gets 50% of CPU and the other two will get 25% of CPU each.

## Memory

# Docker Intelligence.

Knowing your environment will prevent from

docker stats, top, free, cadvisor-influxdb-grafana, systemd-cgtop

# Docker Disaster Prevention.

Docker Containers need justice to survive! They shouldn’t exist without having a leader who knows who works hard and who knows who is running many processes inside.

Easy to do with some parameters

How resources are shared

CGroups

Volumes

Replication Management

Restart Policy

# Conclusion