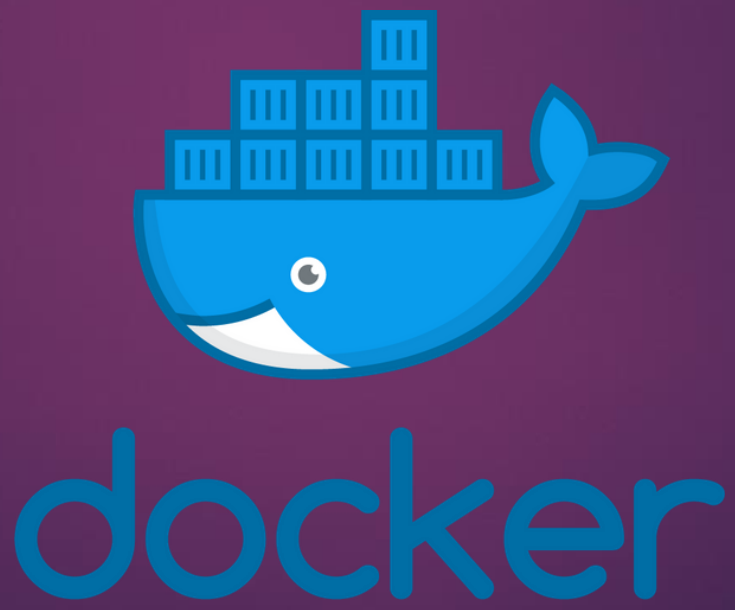
Building Microservices with Kubernetes

BY SHAILEN SUKUL

About

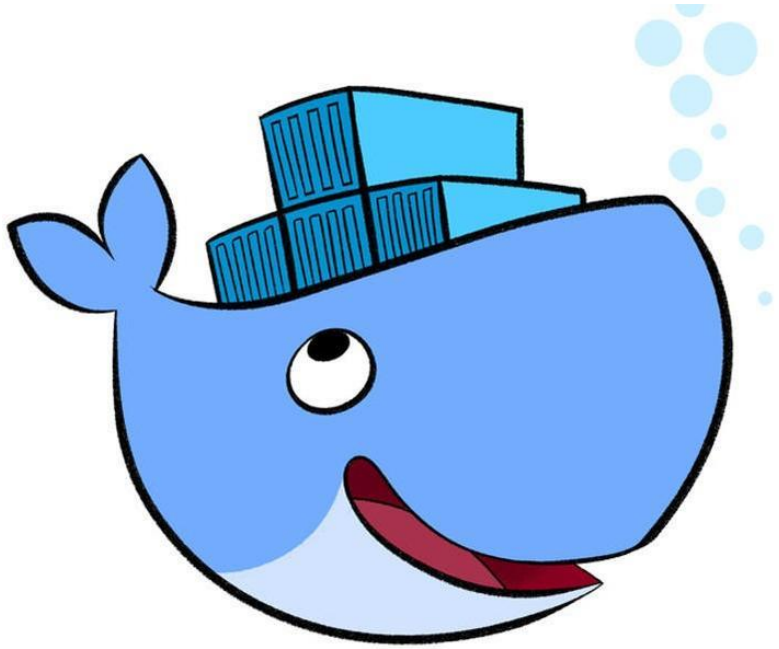
- Azure Cloud, and Office 365 specialist
- Architect, Dev Manager, Team Lead, Dev, BA, PM, DevOps
- www.shailensukul.com
- T: @shailensukul
- Focus on Cloud independence and evangelizing easy to learn Cloud frameworks. Mainly Angular 6, Ionic/Cordova, Web API/.Net Core
- Founder of the **Melbourne South East Cloud Connection** user group

Docker

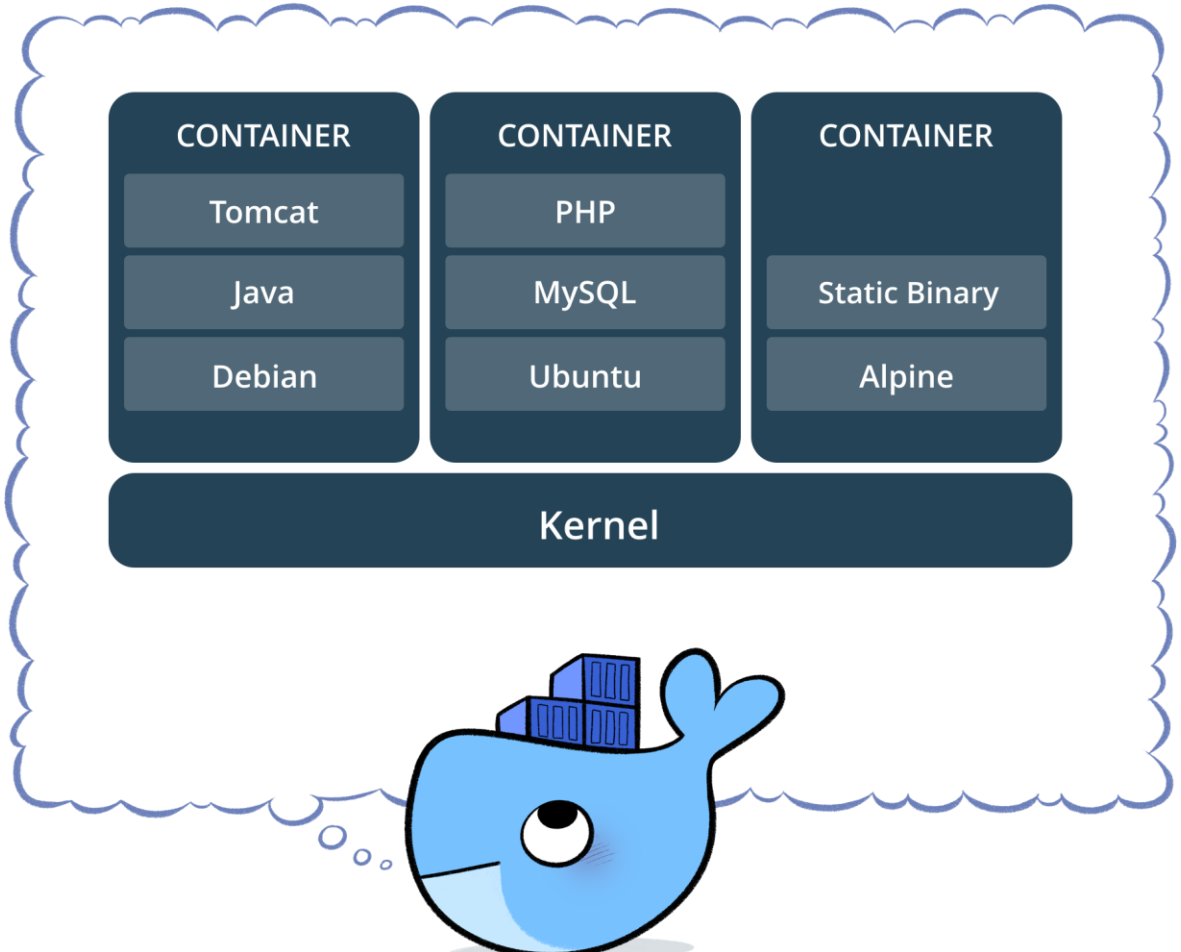


What is Docker?

- Docker enables developers to easily pack, ship, and run any application as a lightweight, portable, self-sufficient container, which can run virtually anywhere.

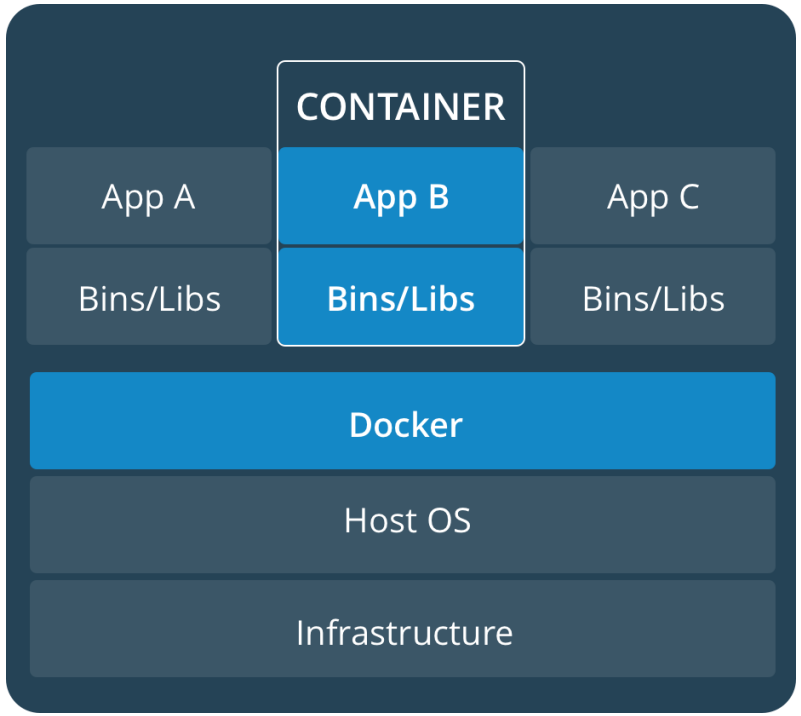


What is a container?

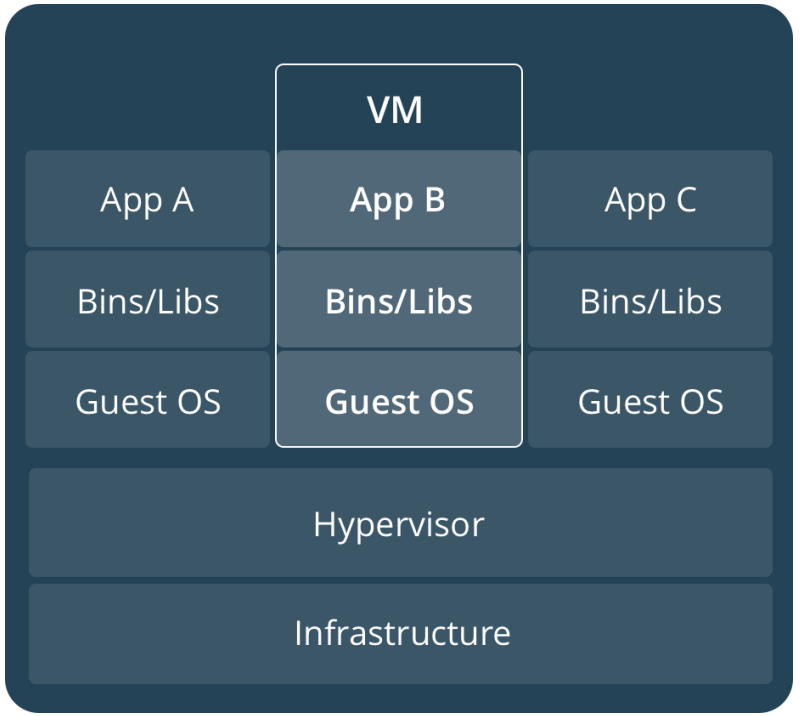


- Standardized packaging for software and dependencies
- Isolate apps from one another
- Share the same OS kernel
- Works with all major Linux and Windows servers

VMs vs Containers

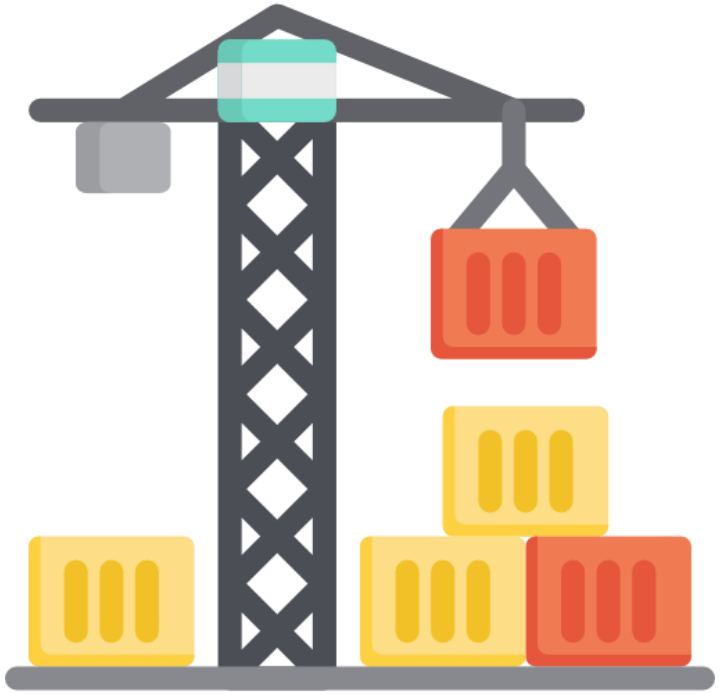


Containers are an App level construct



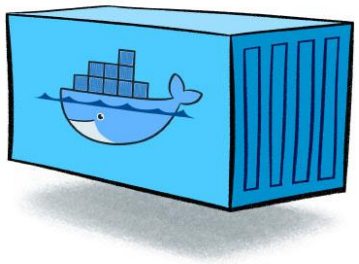
VMs are an infrastructure level construct to turn one machine into many servers

Key Benefits of Docker



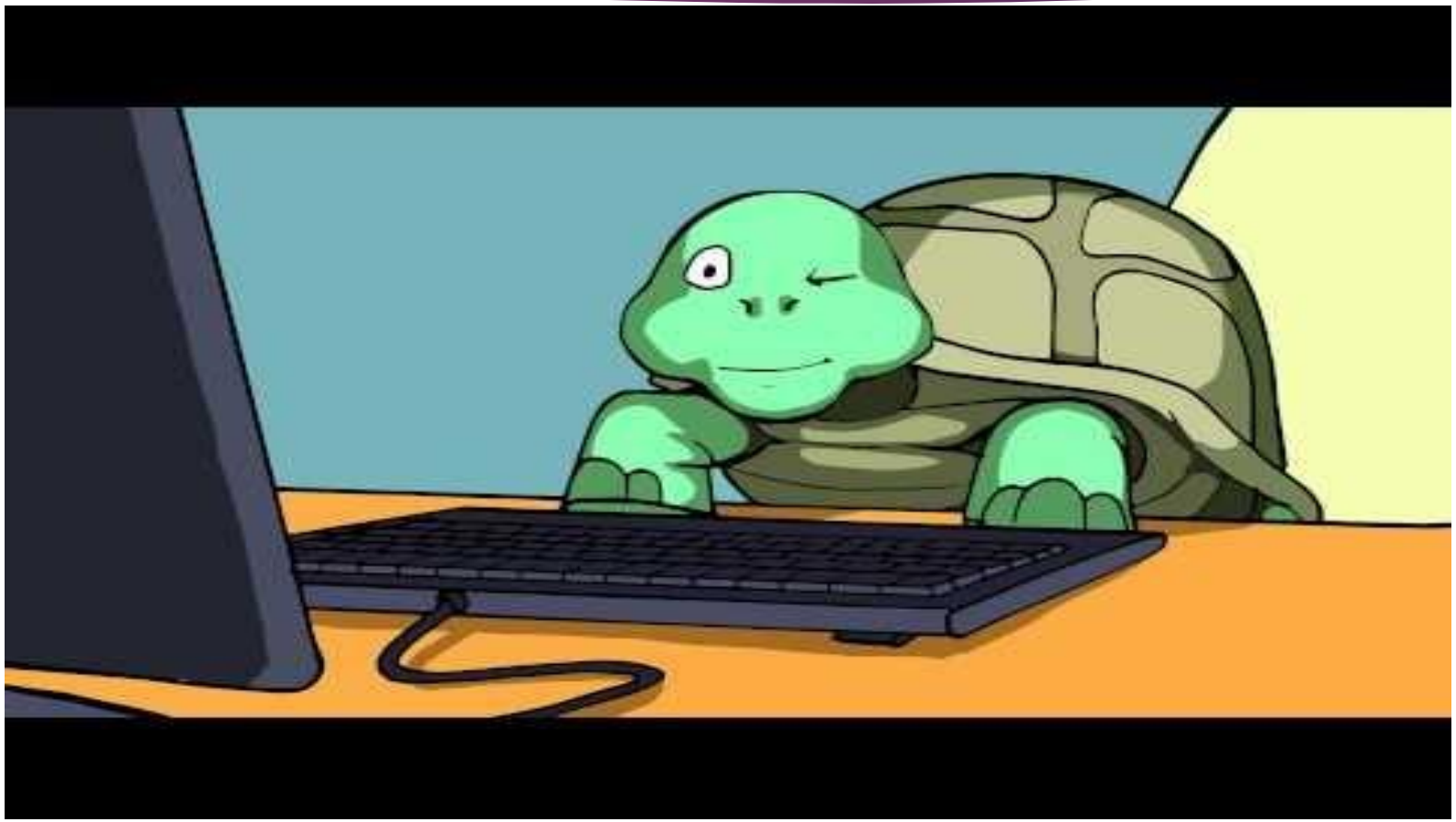
- No OS to boot – application online in seconds
- Portability – Less dependencies between process layers means ability to move between infrastructure
- Efficiency – less OS overhead and improved VM density

Container basics

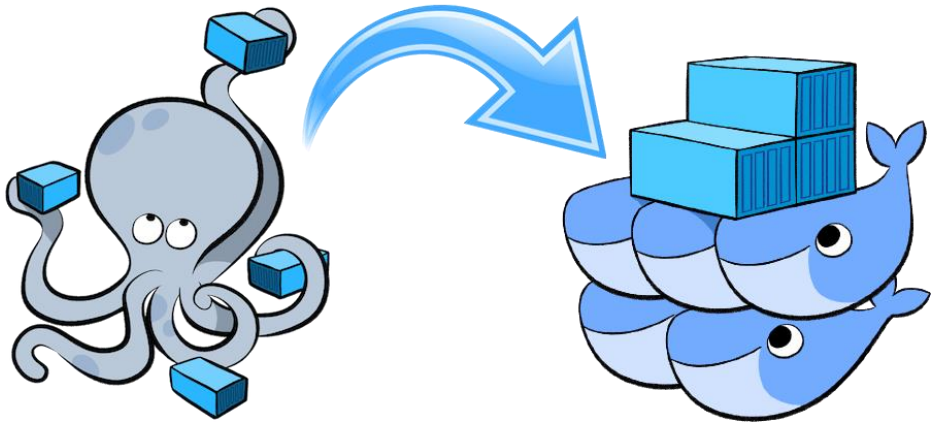


- Image – the basis of a Docker container
- Container – the image when it is running
- Engine – the software that executes commands for containers. Networking and volume are part of engine. Can be clustered together.
- Registry – stores, distributes and managers Docker images
- Control Plane – management plane for container and cluster orchestration

Docker-toon



Docker Compose



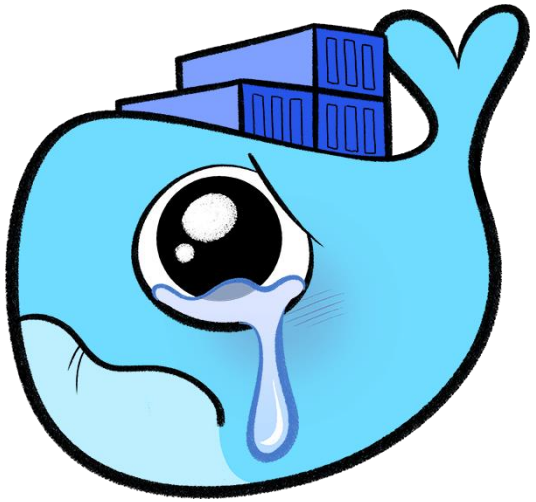
- Compose is a tool for defining and running multi-container Docker applications
- With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

Ok, so why Kubernetes

What Docker does not provide:

- start the right containers at the right time
- figure out how they can talk to each other
- handle storage considerations
- deal with failed containers or hardware

That's where Kubernetes comes in

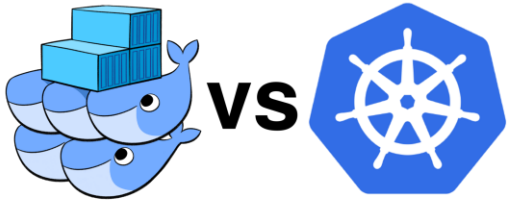


Kubernetes

Kubernetes is an open source container orchestration platform, allowing large numbers of containers to work together in harmony, reducing operational burden. It helps with things like:

- Running containers across many different machines
- Scaling up or down by adding or removing containers when demand changes
- Keeping storage consistent with multiple instances of an application
- Distributing load between the containers
- Launching new containers on different machines if something fails

Kubernetes and Docker

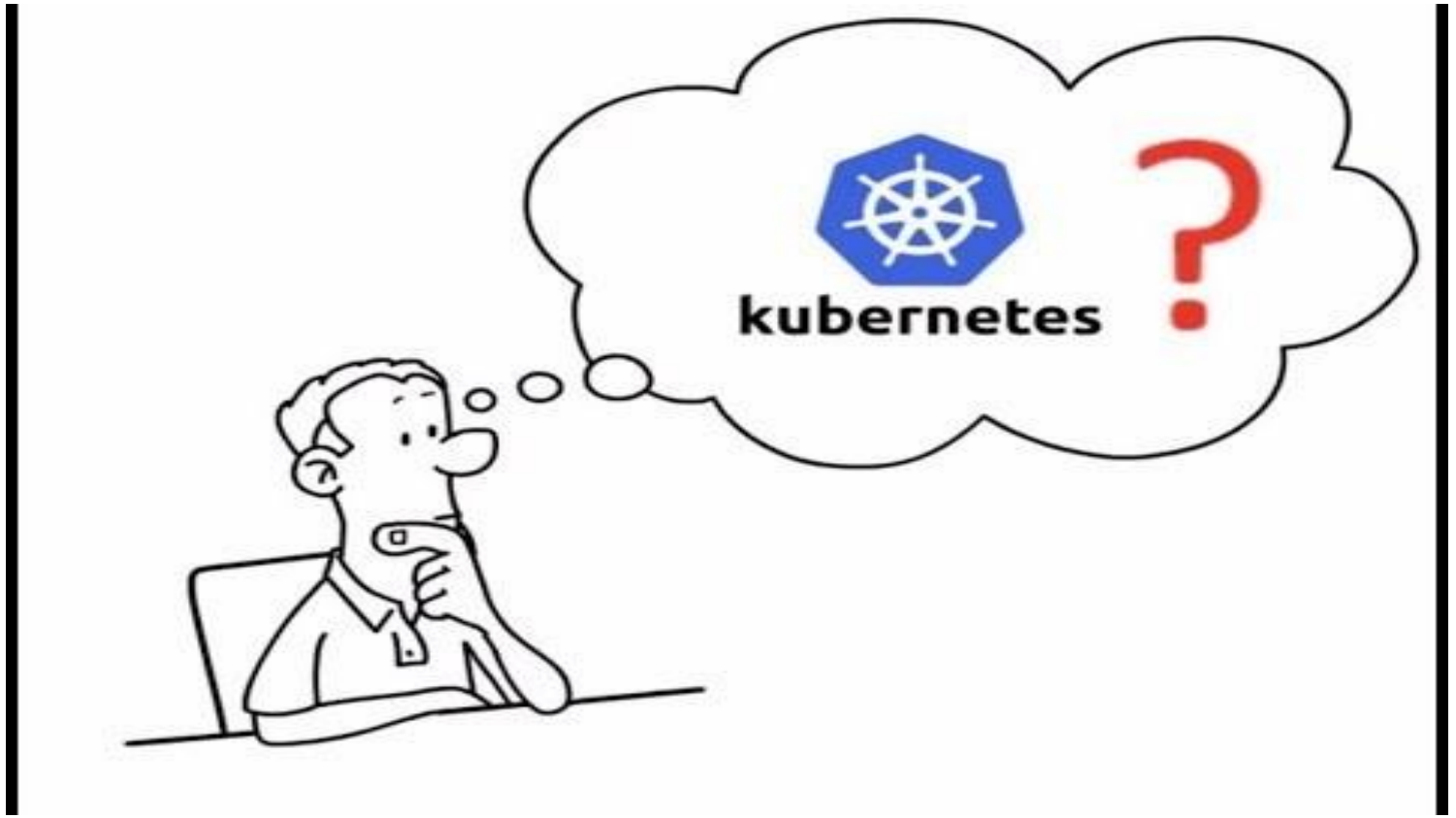


- Docker and Kubernetes work at different levels
- Under the hood, Kubernetes can integrate with the Docker engine to coordinate the scheduling and execution of Docker containers on *Kubelets*.
- The Docker engine itself is responsible for running the actual container image built by running ‘docker build’.
- Higher level concepts such as service-discovery, load balancing and network policies are handled by Kubernetes

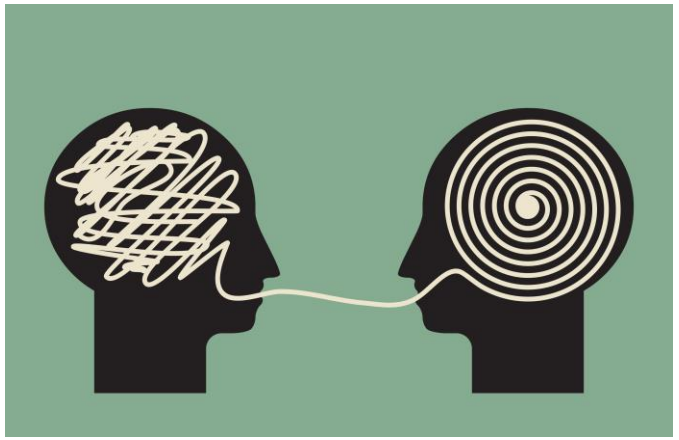
Kubernetes – Cloud Support

- *Managed Support - AWS is the final major provider to add native Kubernetes integration and support. Ostensibly, AWS resisted out of fear for its public cloud business. A cross-platform container manager enables cloud users to set up hybrid container cloud designs, which AWS executives had once dismissed as fake cloud years ago.*
- Portability - Managed Kubernetes services from AWS and Google, as well as the one from Microsoft, are based on open source code, so existing plug-ins, scripts and cluster configurations are portable across platforms.

Kuber-toon



Convert from Compose to Kubernetes



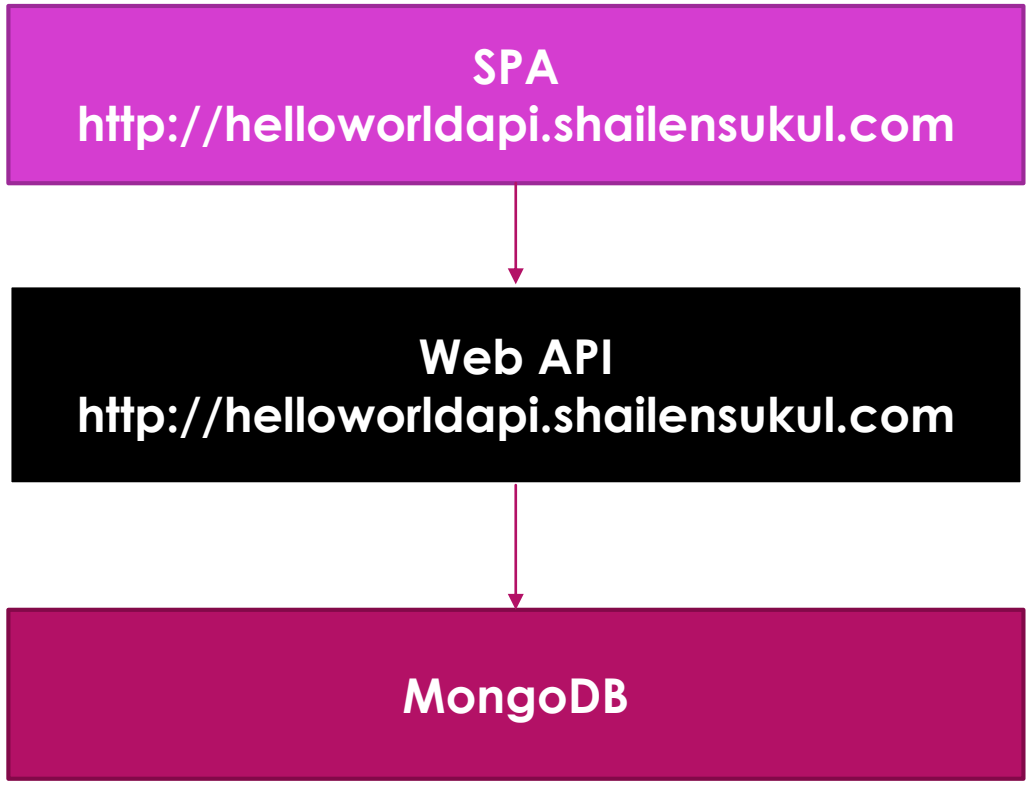
- To work on AKS, the application stack's configuration needs to be reproduced as Kubernetes configuration files.
- Instead of writing the configuration files manually, you can use kompose.
- Kompose is described on its website as '*a conversion tool for Docker Compose to container orchestrators such as Kubernetes.*' Using kompose, I was able to automatically convert the Docker Compose file into analogous Kubernetes resource configuration files.

kompose convert -f docker-compose.yml

Demonstration

Sample Project

- API
- SPA
- Docker Compose file



Sample Project

- Running microservices locally

- docker-compose up
- docker ps
- Docker images at https://hub.docker.com

Sample Project

- Converting docker compose to Kubernetes with Kompose

#install Kompose tool

- choco install kubernetes-komposec
- kompose convert -f docker-compose.yml
- From the original Docker Compose file, containing 3 Docker services, I ended up with 8 individual Kubernetes resource configuration files. Individual configuration files are optimal for fine-grain management of Kubernetes resources

Kompose output

Kubernetes File	
helloworldapi-deployment.yaml	Deployment resource config
helloworldapi-service.yaml	Service config
helloworldspa-deployment.yaml	
helloworldspa-service.yaml	
mongodb-deployment.yaml	
mongodb-service.yaml	
hellowworlddata-persistentvolume.yaml	Created manually Describe the volume
hellowworlddata-persistentvolumeclaim.yaml	Created manually Specify the storage class

Kompose output

- Each Docker service in the Docker Compose file was translated into a separate Kubernetes Deployment resource configuration file, as well as a corresponding Service resource configuration file.
- For the AngularJS Client Service and the API Service, I had to modify the Service configuration files to switch the Service type to a Load Balancer (type: LoadBalancer). Being a Load Balancer, Kubernetes will assign a publicly accessible IP address to each Service

Kompose output

- The MongoDB service requires a persistent storage volume. To accomplish this with Kubernetes, kompose created a PersistentVolumeClaims resource configuration file. I did have to create a corresponding PersistentVolume resource configuration file. It was also necessary to modify the PersistentVolumeClaims resource configuration file, specifying the Storage Class Name as manual, to correspond to the AKS Storage Class configuration

Resources

Url	Notes
https://github.com/shailensukul/Kubernetes-Introduction	Github
http://helloworldazure.shailensukul.com/ http://helloworldgoogle.shailensukul.com/	SPA
http://helloworldapiazure.shailensukul.com/api/values http://helloworldapigoogle.shailensukul.com/api/values	API

Sample Project

- Inspect the Kubernetes files
- Run through the deployment steps
- Show the Kuberentes control panel
- Show the Cnames for the public Ips
- Run through the website

- https://www.youtube.com/watch?v=aqRpZaSX-m8 – hands on in Azure
- https://www.youtube.com/watch?v=NZrYymO6bys- hands on in Google

Thank You!

- Questions?
- Spread the word : https://www.facebook.com/groups/Melbourne.Cloud.Connection

Sponsors

- Officeworks – www.officeworks.com.au
- SharePoint Cloud Design – www.sharepointclouddesign.com