GETTING STARTED WITH KUBERNETES

RAZI RAIS

CONTACT@RAZIBINRAIS.COM

AGENDA

- What are Docker containers?
- Understanding Kubernetes (Architecture/Tools etc.)
- Demo - Build, Deploy , Run a multi-container application using Kubernetes
- Monitoring & Logging in Kubernetes
- Scaling Options in Kubernetes
- Q&A
- Hands on labs

ABOUT NIGEL FRANK INTERNATIONAL

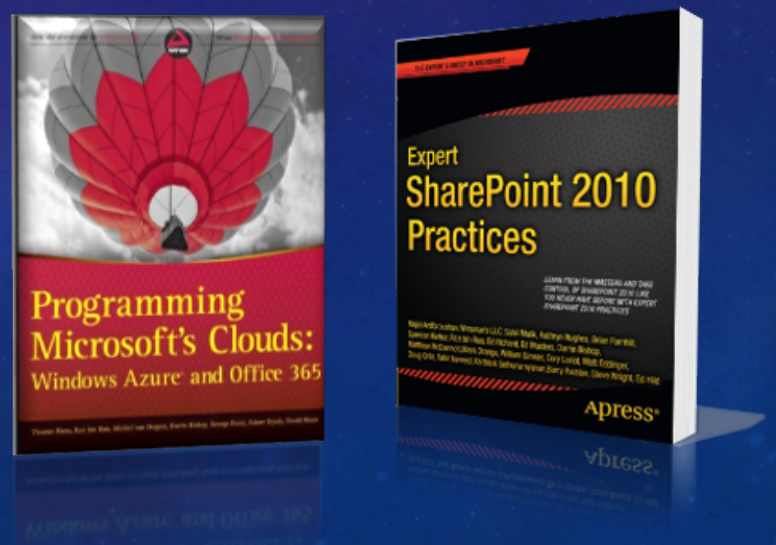
- Founded in 2006 in Newcastle, England, we now have 17 offices globally (soon to be 20), including 8 in the United States
- 800 IT specific recruiters in America alone that cover all the major markets. This includes over 330 in New York, 150 of which focus exclusively on Cloud Technologies.
- Nigel Frank & Jefferson Frank business models that focus on geography and technology specific.
- We are very involved in the community including St Martins School, Women in IT Awards, Tech Academies, Diversity Recruitment Desks and sponsorship of numerous conferences/seminars such as: SF Dreamforce, SN Knowledge, MS Inspire and local user groups

RAZI RAIS

Razi Rais is a senior software engineer at Microsoft with over 15 years of experience in building large scale commercial software and highly scalable cloud services. He is a published author, speaker, trainer on cloud computing, identity, and privacy. He is currently heavily invested in building next-generation event-driven applications using decentralized technologies while preserving end-user privacy.

Podcast: https://bit.ly/2O7qj9J

LinkedIn: https://www.linkedin.com/in/razirais



Microsoft CERTIFIED Trainer

PLURALSIGHT

PLATFORM PROVIDERS

- Google Google Cloud Functions (an fPaaS), Google **Kubernetes** Engine (GKE), Knative (frameworks for building functions and microservices)
- HashiCorp HashiCorp Consul (a service mesh), Nomad (a cluster scheduler)
- Istio Istio (a service mesh for **Kubernetes**)
- Microsoft Microsoft Azure Service Fabric (a microservices framework), Microsoft Azure **Kubernetes** Service (AKS)
- Amazon Web Services AWS Lambda (an fPaaS), Amazon Elastic Container Service (ECS), Amazon Elastic Container Service for Kubernetes (EKS), AWS Fargate, AWS App Mesh (a service mesh)
- Netflix Netflix Common Runtime Services and Libraries (a service mesh)
- Pivotal Pivotal Cloud Foundry (multicloud application platform supporting both Cloud Foundry and **Kubernetes** container systems); also Spring Boot and Spring Cloud Netflix (frameworks for building independently deployable services and for using the Netflix service mesh, respectively)
- Red Hat Red Hat OpenShift Container Platform (a multicloud application platform supporting **Kubernetes**), OpenShift Service Mesh (a service mesh)

Q&A

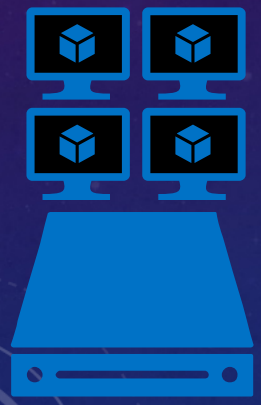
CONTAINERS

A NEW APPROACH TO BUILD, SHIP, DEPLOY, AND INSTANTIATE APPLICATIONS

Physical

Applications traditionally built and deployed onto physical systems with 1:1 relationship

New applications often required new physical systems for isolation of resources



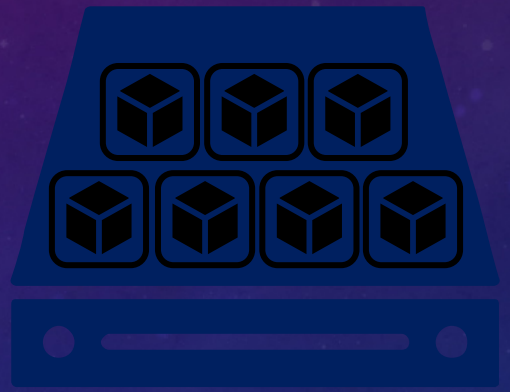
Virtual

Higher consolidation ratios and better utilization

Faster app deployment than in a traditional, physical environment

Apps deployed into VMs with high compatibility success

Apps benefited from key VM features i.e. Live migration, HA



Physical/Virtual

Package and run apps within **Containers**

Key Benefits

Further accelerate of app deployment

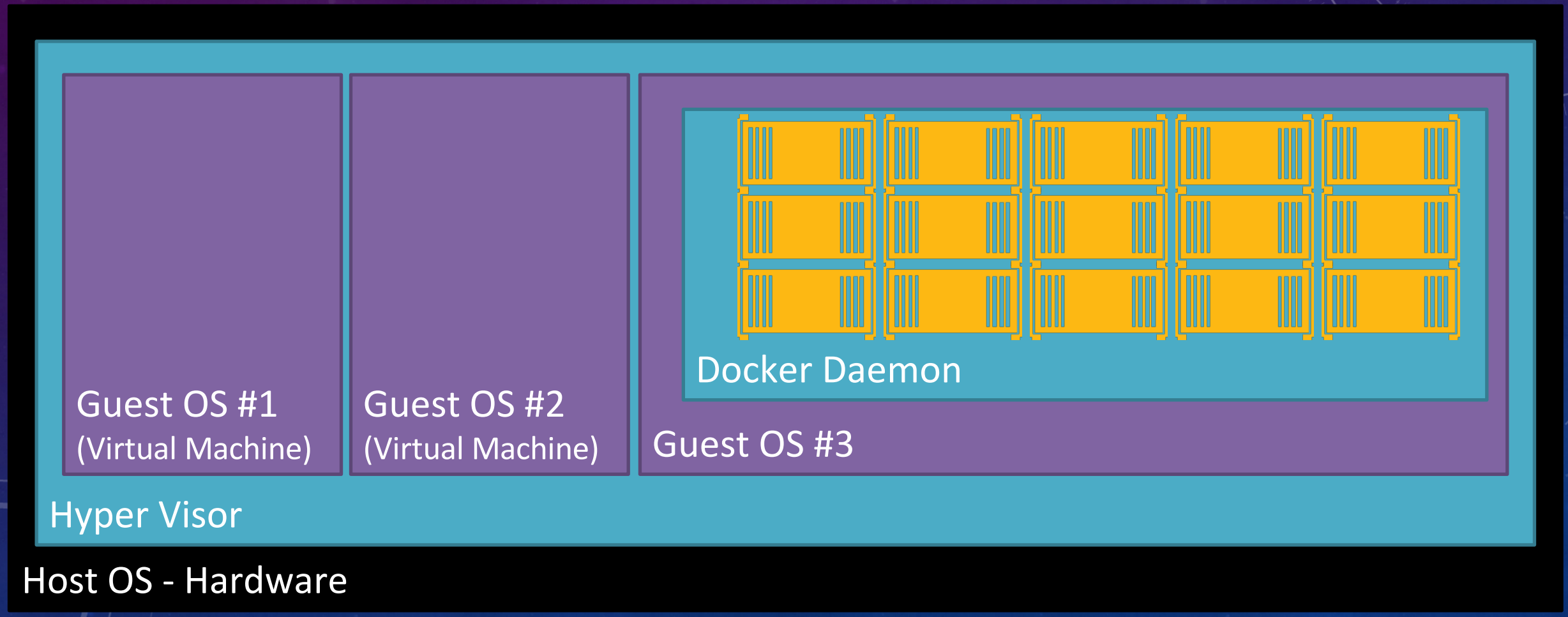
Reduce effort to deploy apps

Streamline development and testing

Lower costs associated with app deployment

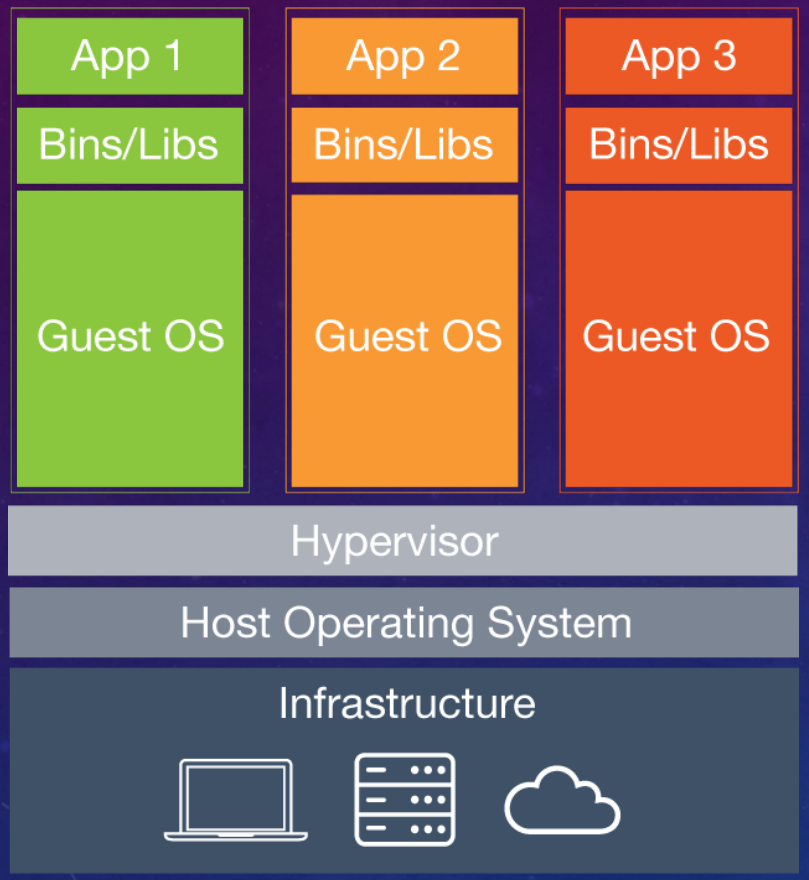
Increase server consolidation

DOCKER CONTAINERS AND VMS

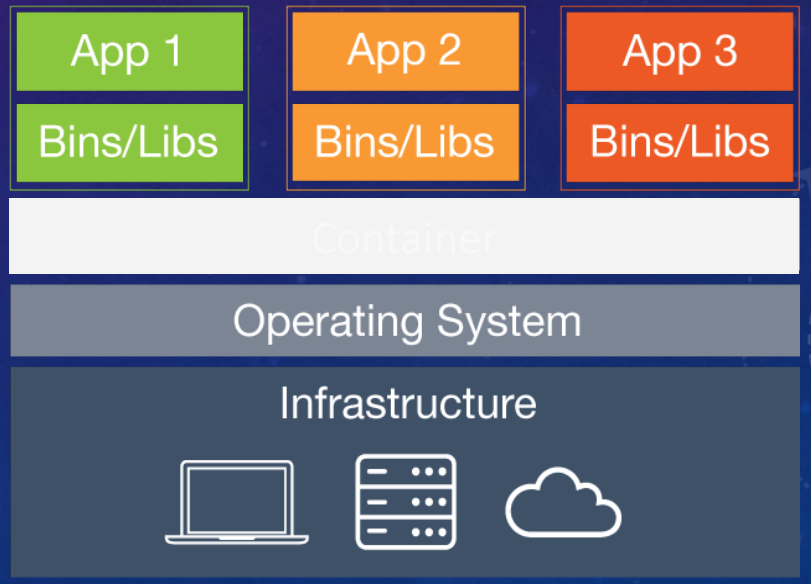


CONTAINERS ARE SIGNIFICANTLY MORE LIGHTWEIGHT THAN A VM.

Virtual Machine



Container



KUBERNETES HISTORY

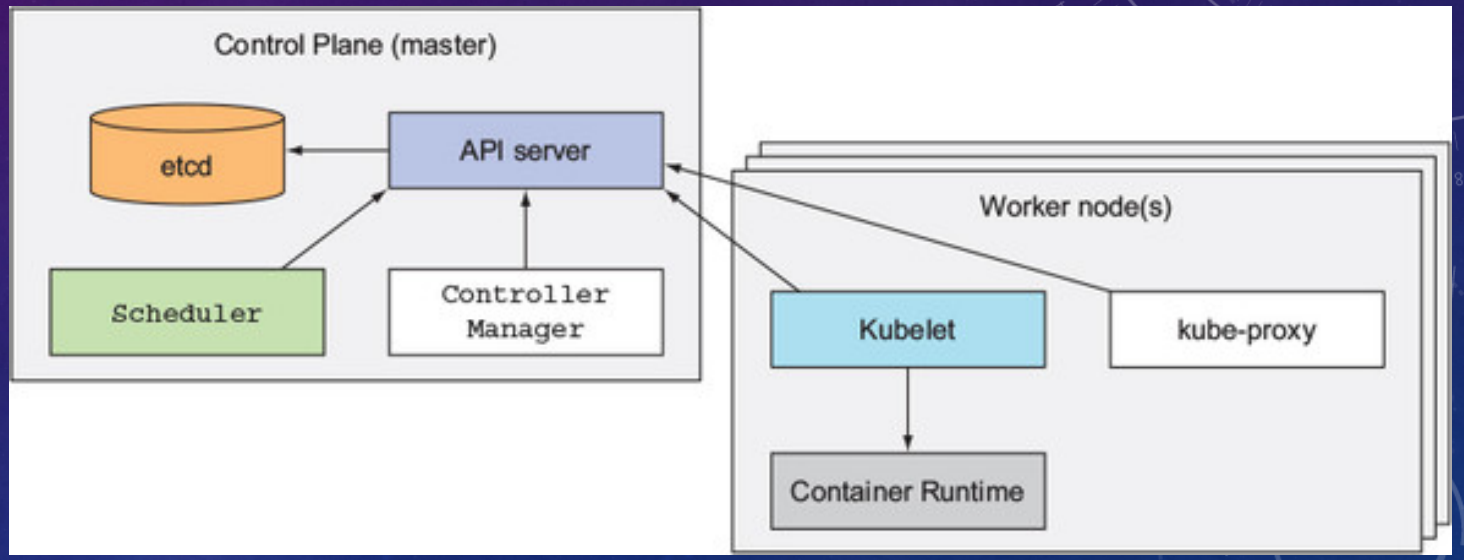
- Heavily influenced by Google's Borg system
- Originally created by Google and donated to the Cloud Native Computing Foundation (CNCF) in 2015
- Original codename was Project Seven, the seven spokes on the Kubernetes logo is a nod that codename
- Kubernetes is Greek for Helmsman or Captain

ORCHESTRATION != CONTAINERS

- Containers are:
  - a unit of repeatable deployment
  - a security and isolation boundary
- Orchestration is:
  - Automatic Failover
  - Monitored Upgrades
  - State Management
  - Resource Monitoring and Management
  - Constraints
  - Health Checks
  - More…

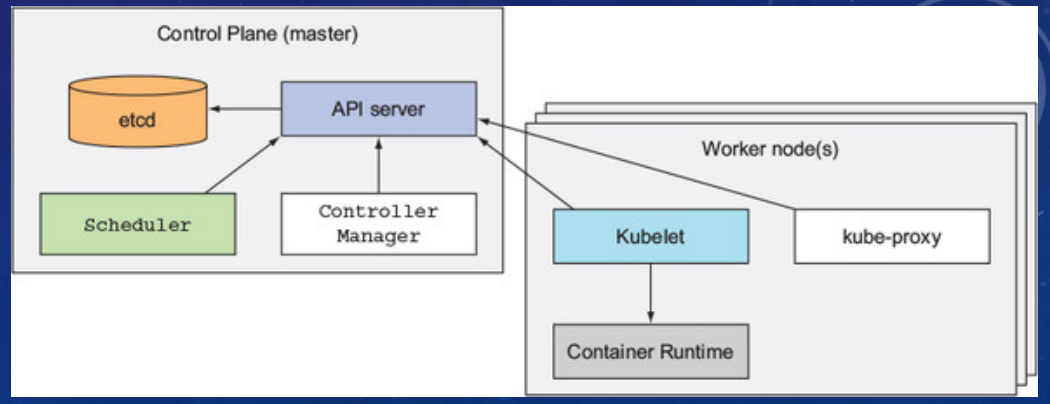
KUBERNETES ARCHITECTURE | BIRD'S EYE VIEW

- The *Master node*, which hosts the *Kubernetes Control Plane* that controls and manages the whole Kubernetes system
- *Worker nodes* that run the actual applications you deploy



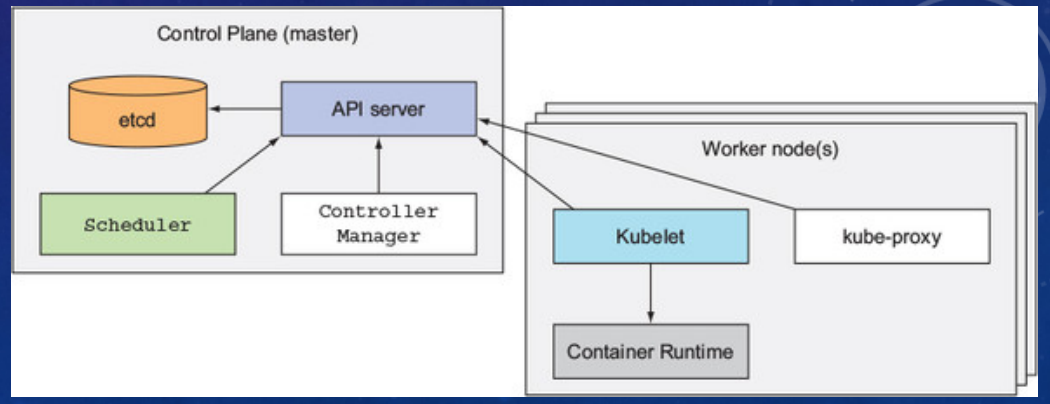
KUBERNETES ARCHITECTURE | MASTER OR CONTROL PANE

- The **Control Plane** is what controls the cluster and makes it function. It consists of multiple components that can run on a single master node or be split across multiple nodes and replicated to ensure high availability. These components are
- The **Kubernetes API Server**, which you and the other Control Plane components communicate with
- The **Scheduler**, which schedules your apps (assigns a worker node to each deployable component of your application)
- The **Controller Manage**r, which performs cluster-level functions, such as replicating components, keeping track of worker nodes, handling node failures, and so on
- **etcd**, a reliable distributed data store that persistently stores the cluster configuration.



KUBERNETES ARCHITECTURE | WORKER NODE(S)

- The **worker nodes** are the machines that run your containerized applications. The task of running, monitoring, and providing services to your applications is done by the following components:-
- **Docker, rkt,** or another **container runtime**, which runs your containers
- The **Kubelet**, which talks to the API server and manages containers on its node
- The **Kubernetes Service Proxy (kube-proxy)**, which load-balances network traffic between application components



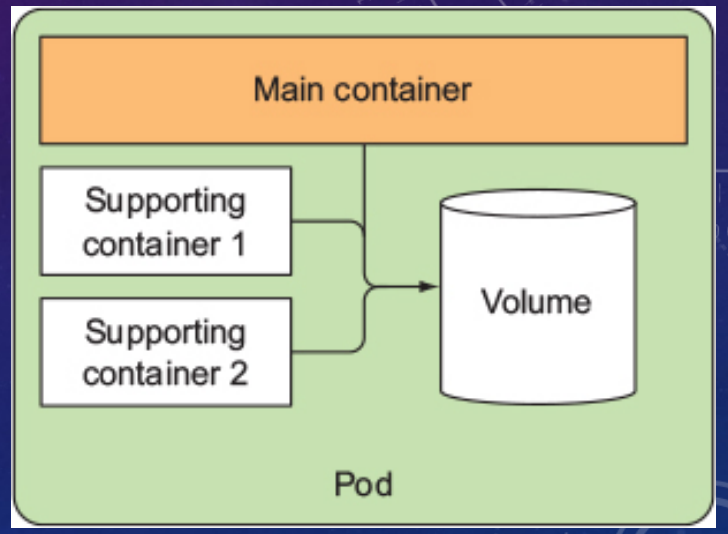
KUBERNETES TOOLS | MINIKUBE + KUBECTL

- Minikube offers a complete test environment that runs on Linux, OS-X or Windows
- Minikube as it is easy to setup, open source and has no further dependencies
- Kubectl is a command line interface (CLI) for running commands against Kubernetes clusters

DEMO

PODS

- Pod is a co-located group of containers and represents the basic building block in Kubernetes.
- Instead of deploying containers individually, you always deploy and operate on a pod of containers.
- Pod contents are always co-located and co-scheduled, and run in a shared context. A Pod models an application-specific “logical host” - it contains one or more application containers which are relatively tightly coupled
- Pod can have a single container (as shown in the next demo)

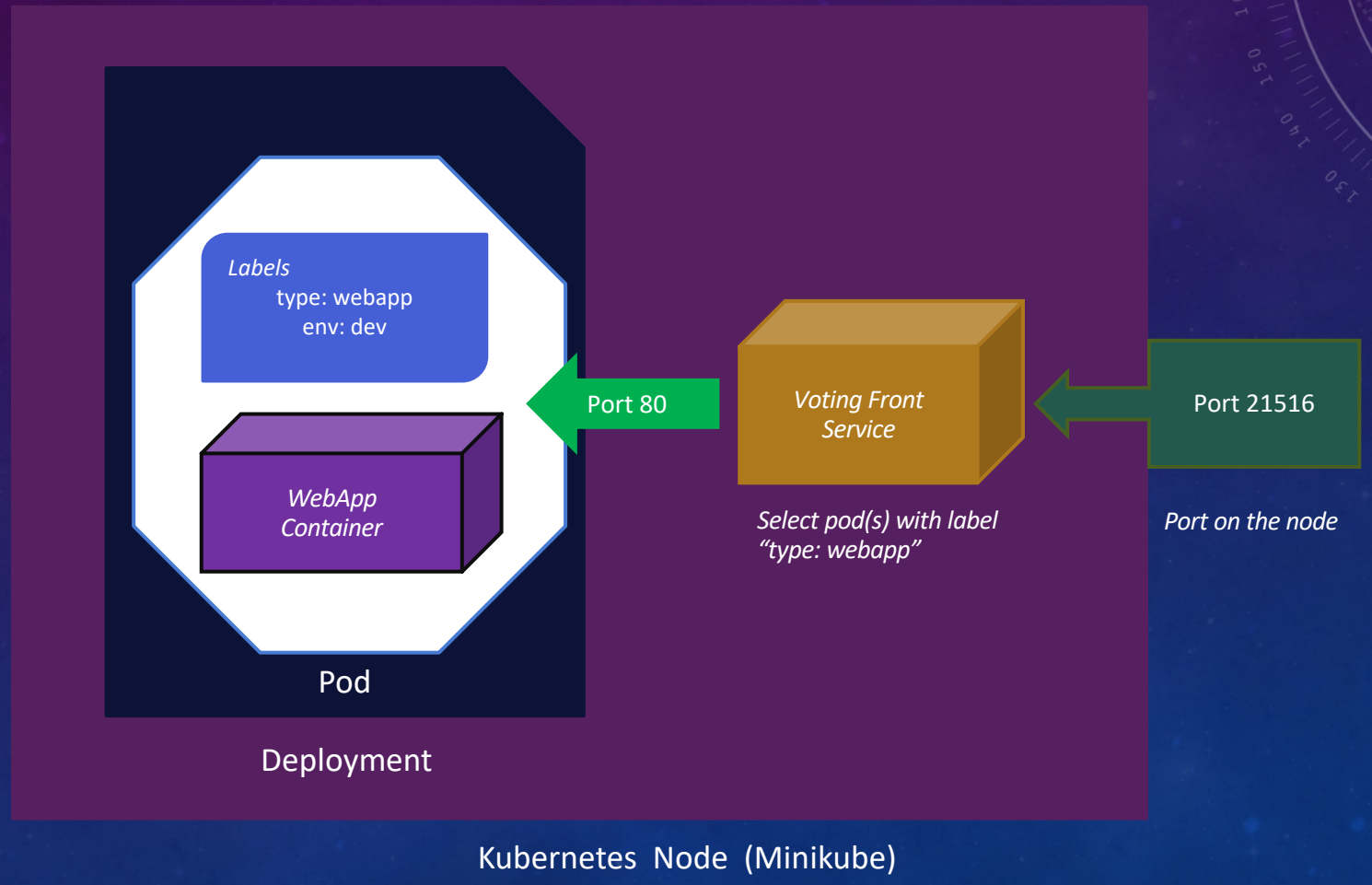


Pods should contain tightly coupled containers, usually a main container and containers that support the main one.

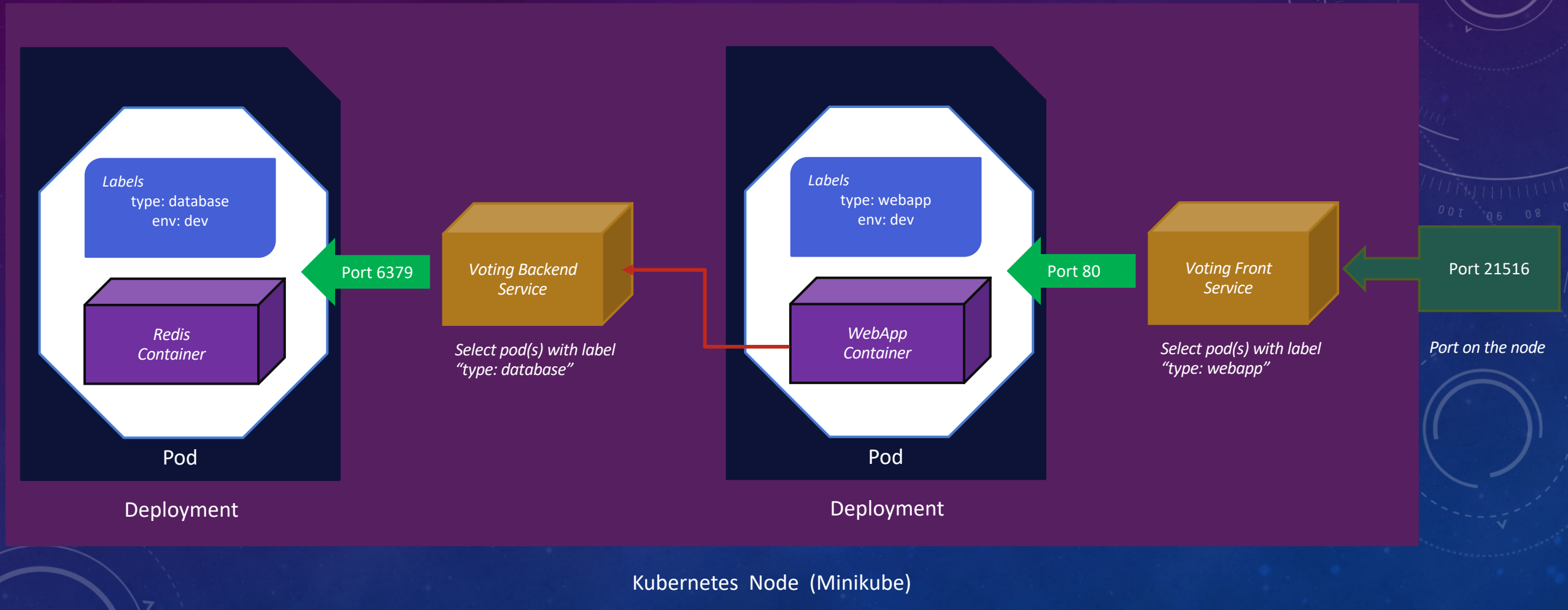
REPLICASET & DEPLOYMENT

- The pod is the most basic entity in Kubernetes
- To determine how many instances of a pod you want to run, you need replication sets
- Deployments are used to create and automate replica sets
- Deployments instruct the cluster how to create and scale applications
- The deployment controller will monitor instances of an application
- Use deployments to create replica sets
- Deployments allow for the creation of multiple replica sets for rolling upgrades or rollbacks

KUBERNETES | VOTING APPLICATION FRONT END WEBAPP



KUBERNETES | VOTING APPLICATION FRONT END & BACKEND



SCALE UP/DOWN THE DEPLOYMENT TO FACILITATE MORE LOAD.

- To meet the traffic needs you can scale up or down a deployment
- `kubectl scale deployment-name --replicas={number-of-desired-pods}`
- Behind the scenes Kubernetes will increase/decrease the replica set count to ensure that only desired number of pods are running

DEMO

ROLLOUT DEPLOYMENT WITH ZERO DOWNTIME

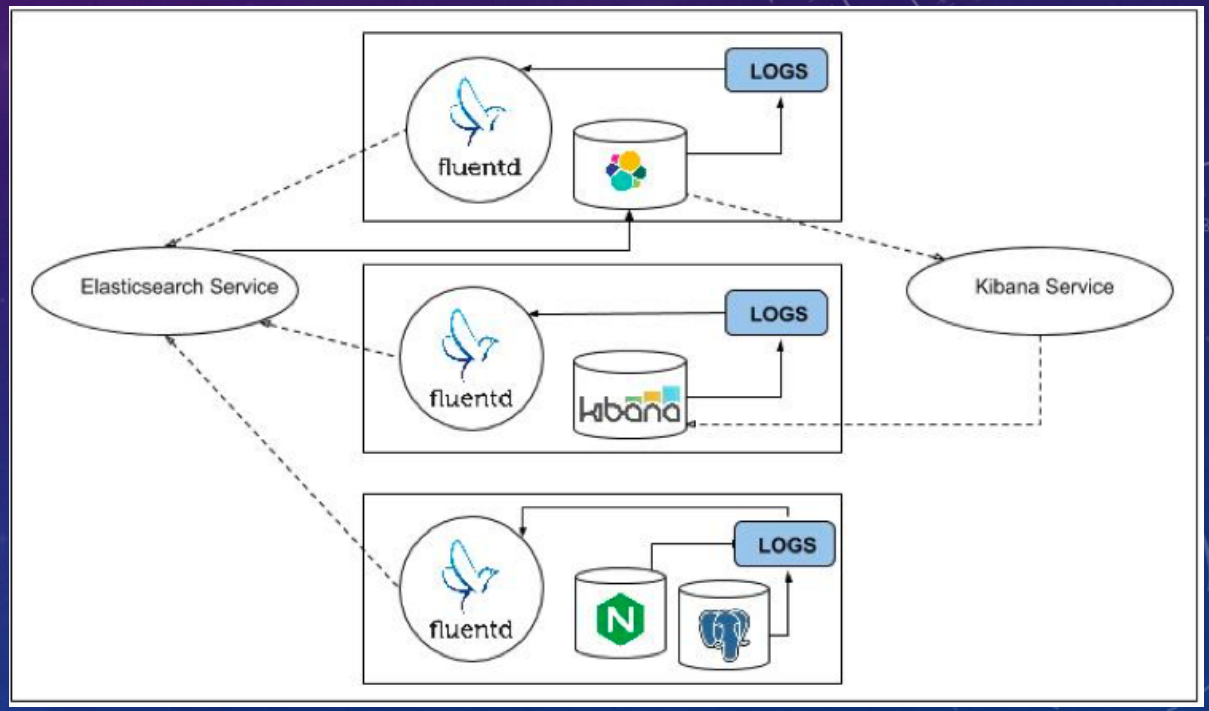
- Kubernetes supports deployment strategy *Rolling Update* that allow update to take place with zero downtime by incrementally updating Pods instances with new ones.
- You can define max number (`maxUnavailable`) of pods that can be unavailable during deployment. By default its one.
- You can also define maximum number of new Pods that can be created (`maxSurge`). By default its one.

```
strategy:
  rollingUpdate:
    maxSurge: 1
    maxUnavailable: 1
```

DEMO

LOGGING

- Everything a containerized application writes to stdout and stderr is handled by container engine
- `kubectl logs` command give you access to container logs running within a Pod
- Advanced solutions are needed to get holistic logging support
- EFK (Elasticsearch, Fluentd and Kibana) is a powerful tool
- Elasticsearch is a search engine that is responsible for storing our logs and allowing for them to be queried.
- Fluentd sends log messages from Kubernetes to Elasticsearch
- Kibana is a graphical interface for viewing and querying the logs stored in Elasticsearch.



DEMO

MONITORING

- Kubernetes cluster performance can be examined by looking at the containers, pods, services, and the characteristics of the overall cluster.
- Prometheus can natively monitor Kubernetes, nodes, and prometheus itself.
- Prometheus Operator simplifies Prometheus setup on Kubernetes, and allows you to serve the custom metrics API using the Prometheus adapter
  Sysdig
- Sysdig provides full spectrum container and platform intelligence, and is a true container native solution.
- Sysdig pulls together data from system calls, Kubernetes events, Prometheus metrics, statsD, JMX, and more into a single pane that gives you a comprehensive picture of your environment.
- Sysdig also provides an API to query for providing robust and customizable solutions.

DEMO

HANDS-ON LAB

1. Installation (Minikube/Docker)
2. Build, Deploy and Run | Multi-Container Application
3. Scaling | Scale web front app
4. Rolling Update & Rollback | Perform update with zero downtime strategy
5. Monitoring (Grafana) | Use Grafana dashboard for monitoring
6. Logging (EFK) | Use Elasticsearch, Fluentd and Kinbana (EFK) for logging

https://github.com/razi-rais/microservices/blob/master/workshop/readme.md

THANK YOU!