DEV CONTAINERS

FROM DEVELOPMENT TO DEPLOYMENT

Venkatt Guhesan
Dec. 2022

Our journey so far

Two Variants of Dev Containers and Code/Install Isolation

- ❑ NixOS.org
- ❑ Dev Containers and using VSC Templates and "Dev Container" extension as a starting point for our Dev Containers

Using the Dev Container extension, we looked at samples to auto-generate "docker-compose.yaml" as part of the workspace.

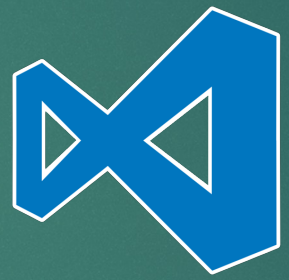
Looked into converting a "docker-compose.yaml" file to a collection of Kubernetes configuration files using Kompose.

We then took the collection of Kubernetes configuration files to assemble them into a config document to generate a single Kubernetes manifest file using Kustomize.

Our Development Environment

With Kubernetes enabled

+



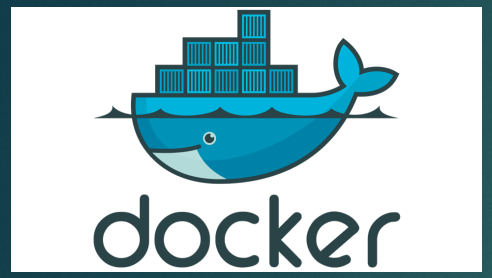
Visual Studio Code

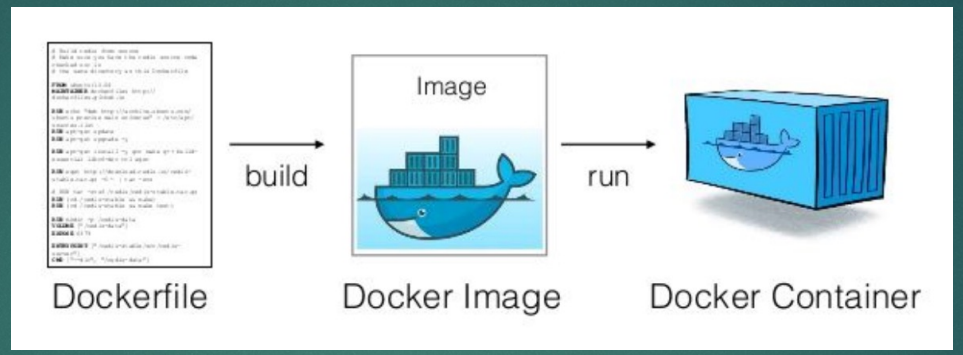
+

Dev Containers

Cross-Platform Solution
Works the same in Windows, Mac and Linux

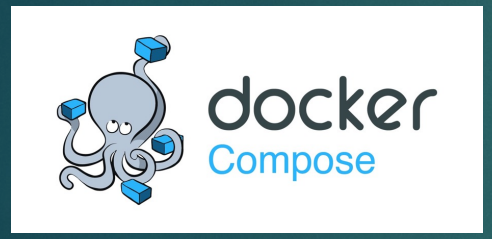
Docker vs. Docker Compose

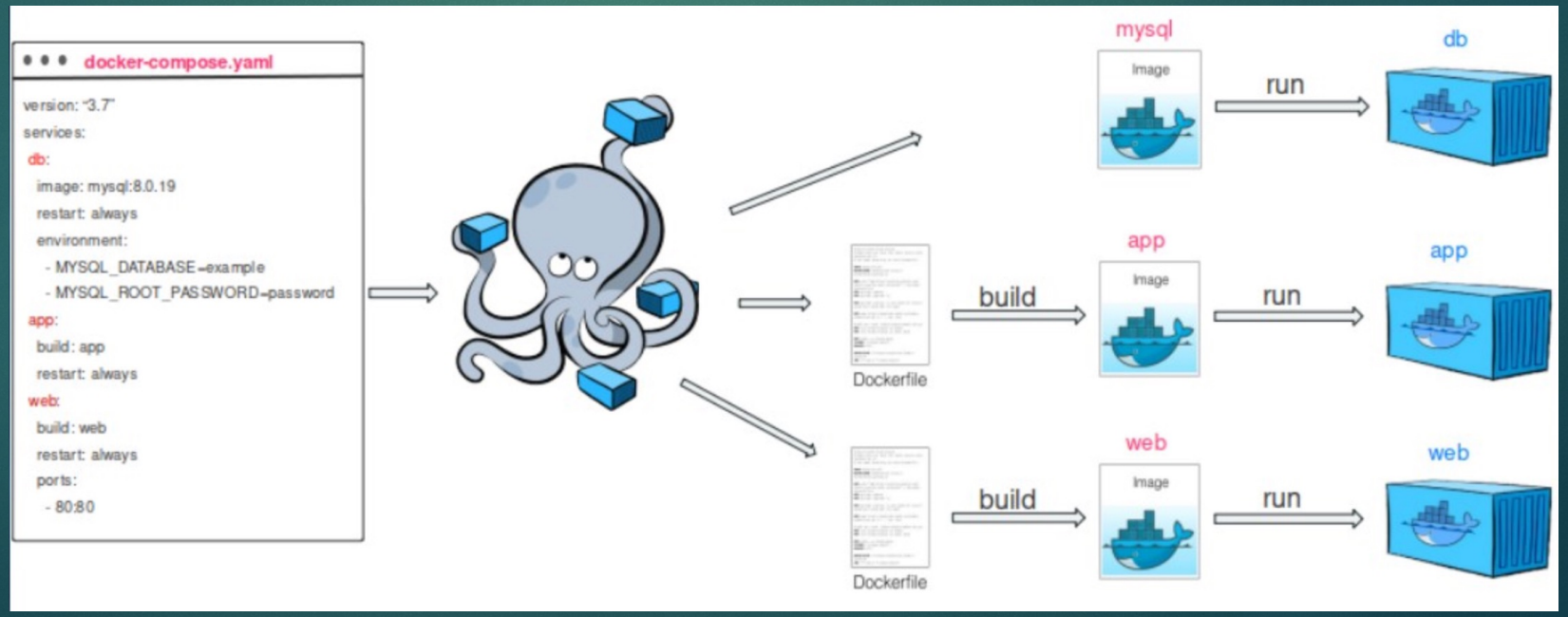




Quick Links

- ✓ Docker vs Docker Compose





Docker Compose v2 Format

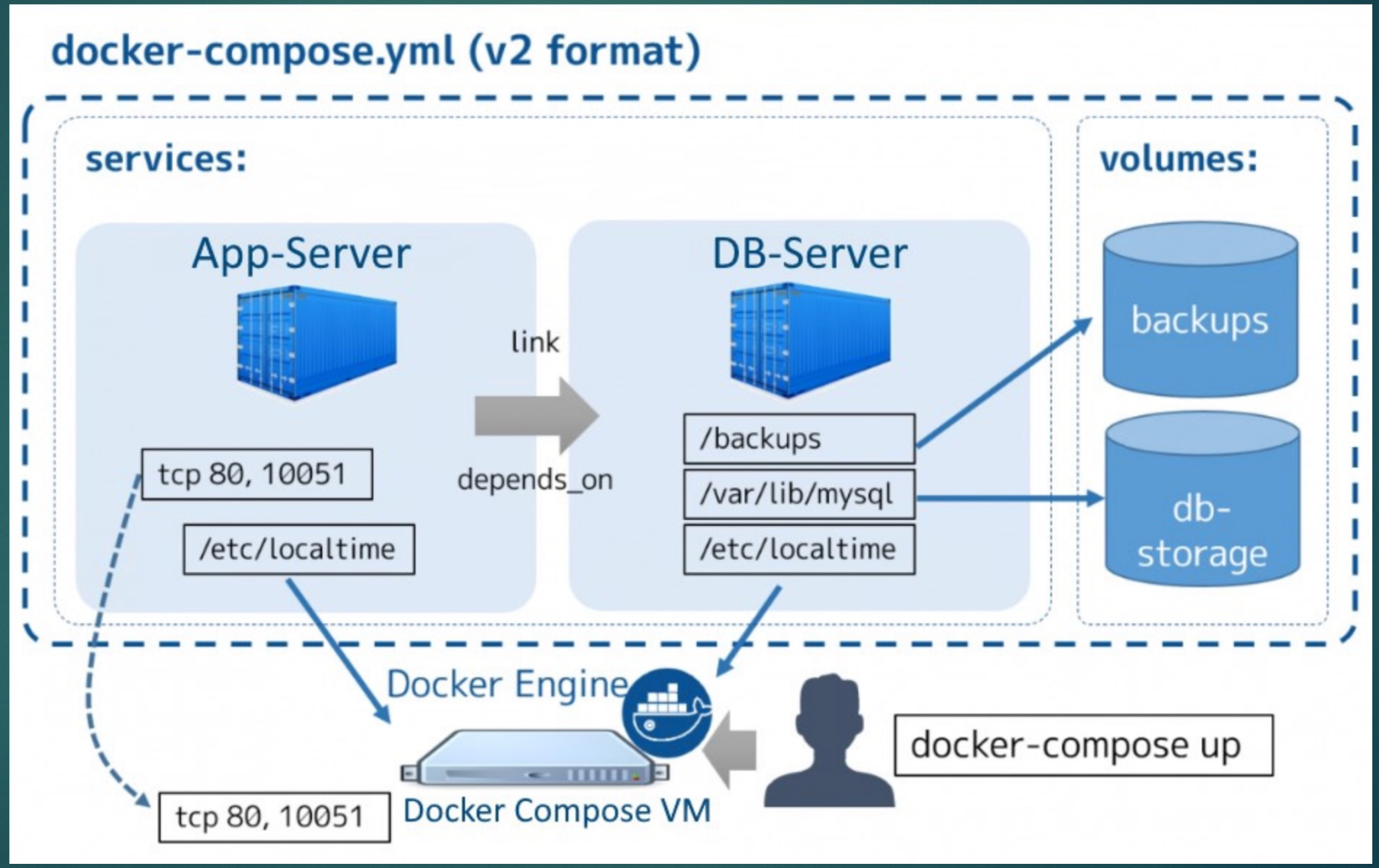
docker Compose

Docker Compose is a private-label of Fig by Orchard Up

Compose File:

- compose.yaml
- compose.yml
- docker-compose.yaml
- docker-compose.yml

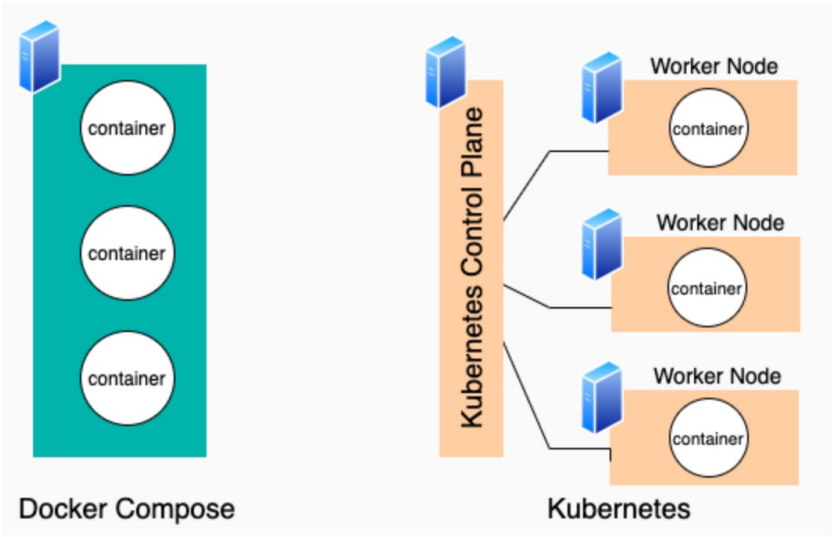
Learn more…

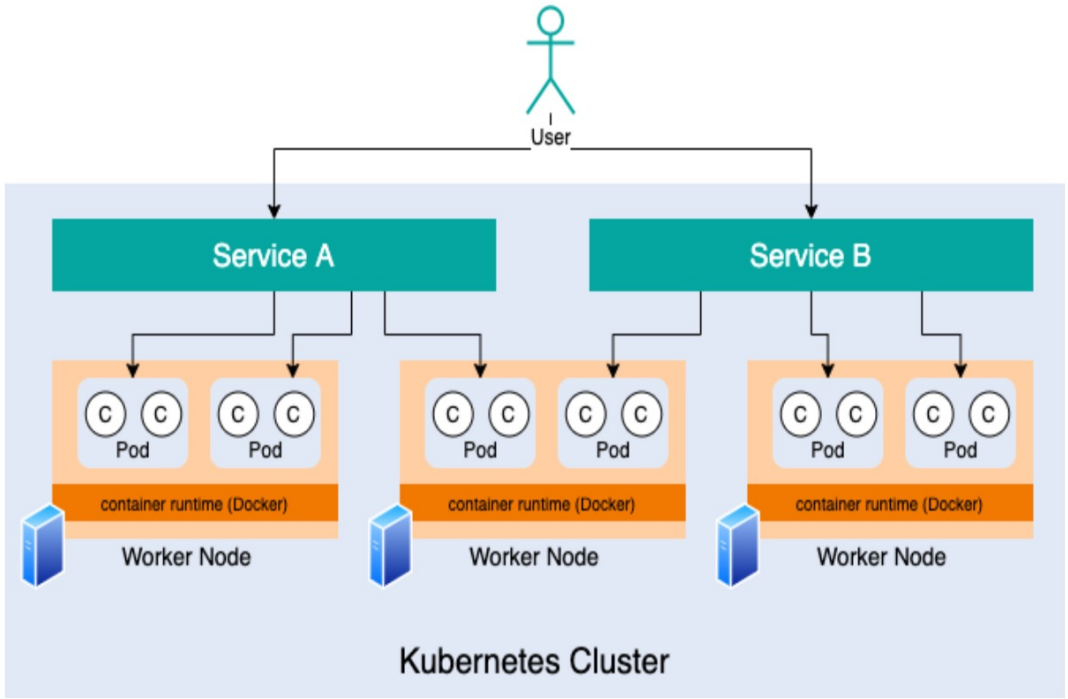


Docker-Compose is Docker's flavor of Kubernetes Orchestration

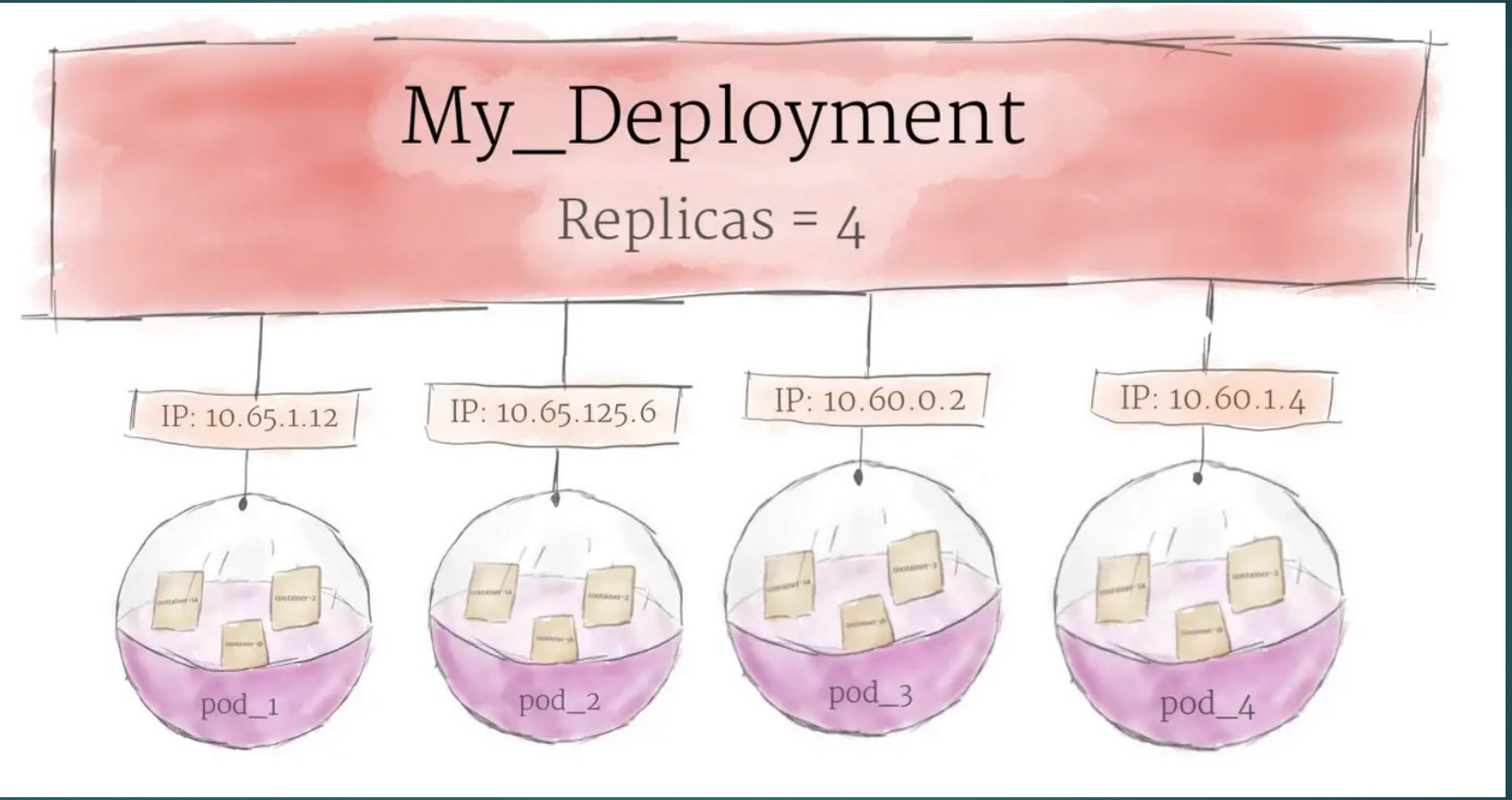
Docker Compose vs Kubernetes

The main difference is that Kubernetes runs containers across multiple computers, whether virtual or physical, whereas Docker Compose runs containers on a single host machine (for our development needs).





Pod in detail



credit

K8s cont.

Pod

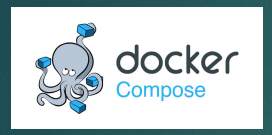
- Pod is the basic unit of work and the smallest deployable unit of computing that you can create and manage in Kubernetes.
- Pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers.
- Pod is a thin wrapper around one or more containers.

Namespace

- Namespaces provides a mechanism for isolating groups of resources within a single cluster.
- Names of resources need to be unique within a namespace, but not across namespaces.
- Namespace-based scoping is applicable only for namespaced objects *(e.g. Deployments, Services, etc)* and not for cluster-wide objects *(e.g. StorageClass, Nodes, PersistentVolumes, etc)*.
- Namespaces enable organizing resources into non-overlapping groups (for example, per tenant, per environment, per project, per team)

Learn more . . .

Docker & Kubernetes Key Differences



docker-compose.yaml

```
version: '3'
services:
  web:
    build: .
    ports:
     - "8080:80"
  db:
    image: mysql
    ports:
    - "3306:3306"
    environment:
    - MYSQL_ROOT_PASSWORD=password
    - MYSQL_USER=user
    - MYSQL_PASSWORD=password
    - MYSQL_DATABASE=demodb
```

Kubernetes Deployment Template

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Docker & Kubernetes Key Differences

Docker Compose Application Model

- Services
- Networks
- Volumes
- Configs
- Secret

Kubernetes Objects/Resources/Kinds

```
$ kubectl api-resources
```

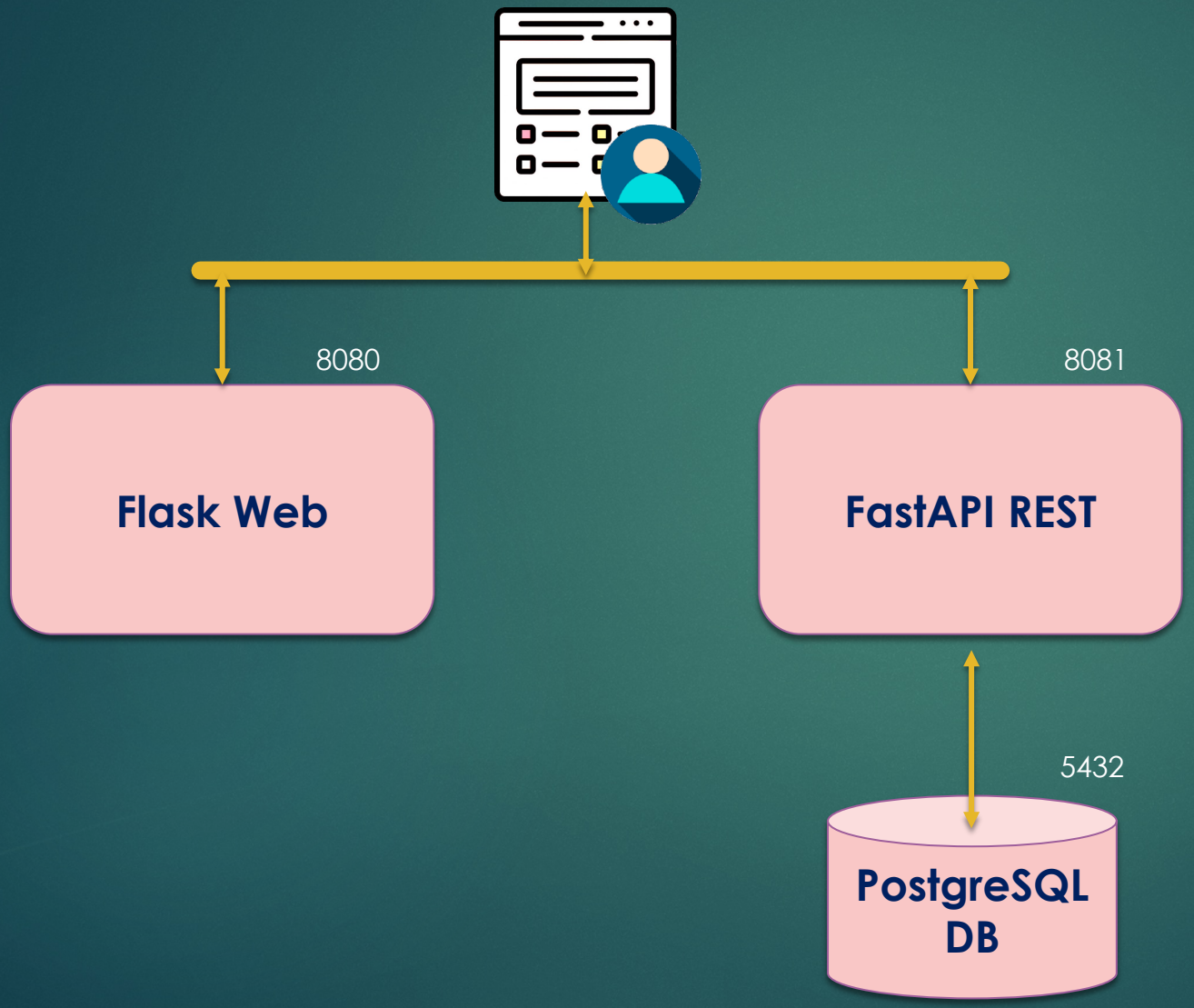
NAME	SHORTNAMES	KIND
bindings		Binding
componentstatuses	cs	ComponentStatus
configmaps	cm	ConfigMap
endpoints	ep	Endpoints
events	ev	Event
limitranges	limits	LimitRange
namespaces	ns	Namespace
nodes	no	Node
persistentvolumeclaims	pvc	PersistentVolumeClaim
persistentvolumes	pv	PersistentVolume
pods	po	Pod
podtemplates		PodTemplate
replicationcontrollers	rc	ReplicationController
resourcequotas	quota	ResourceQuota
secrets		Secret
serviceaccounts	sa	ServiceAccount
services	svc	Service
initializerconfigurations		InitializerConfiguration
mutatingwebhookconfigurations		MutatingWebhookConfiguration
validatingwebhookconfigurations		ValidatingWebhookConfiguration
customresourcedefinitions	crd,crds	CustomResourceDefinition
apiservices		APIService
controllerrevisions		ControllerRevision
daemonsets	ds	DaemonSet
deployments	deploy	Deployment
replicasets	rs	ReplicaSet
statefulsets	sts	StatefulSet
tokenreviews		TokenReview
localsubjectaccessreviews		LocalSubjectAccessReview
selfsubjectaccessreviews		SelfSubjectAccessReview
selfsubjectrulesreviews		SelfSubjectRulesReview
subjectaccessreviews		SubjectAccessReview
horizontalpodautoscalers	hpa	HorizontalPodAutoscaler
cronjobs	cj	CronJob
jobs		Job
brpolices	br,bp	BrPolicy
clusters	rcc	Cluster

NAME	SHORTNAMES	KIND
filesystems	rcfs	Filesystem
objectstores	rco	ObjectStore
pools	rcp	Pool
certificatesigningrequests	csr	CertificateSigningRequest
leases		Lease
events	ev	Event
daemonsets	ds	DaemonSet
deployments	deploy	Deployment
ingresses	ing	Ingress
networkpolicies	netpol	NetworkPolicy
podsecuritypolicies	psp	PodSecurityPolicy
replicasets	rs	ReplicaSet
nodes		NodeMetrics
pods		PodMetrics
networkpolicies	netpol	NetworkPolicy
poddisruptionbudgets	pdb	PodDisruptionBudget
podsecuritypolicies	psp	PodSecurityPolicy
clusterrolebindings		ClusterRoleBinding
clusterroles		ClusterRole
rolebindings		RoleBinding
roles		Role
volumes	rv	Volume
priorityclasses	pc	PriorityClass
storageclasses	sc	StorageClass
volumeattachments		VolumeAttachment

Our Next Objective

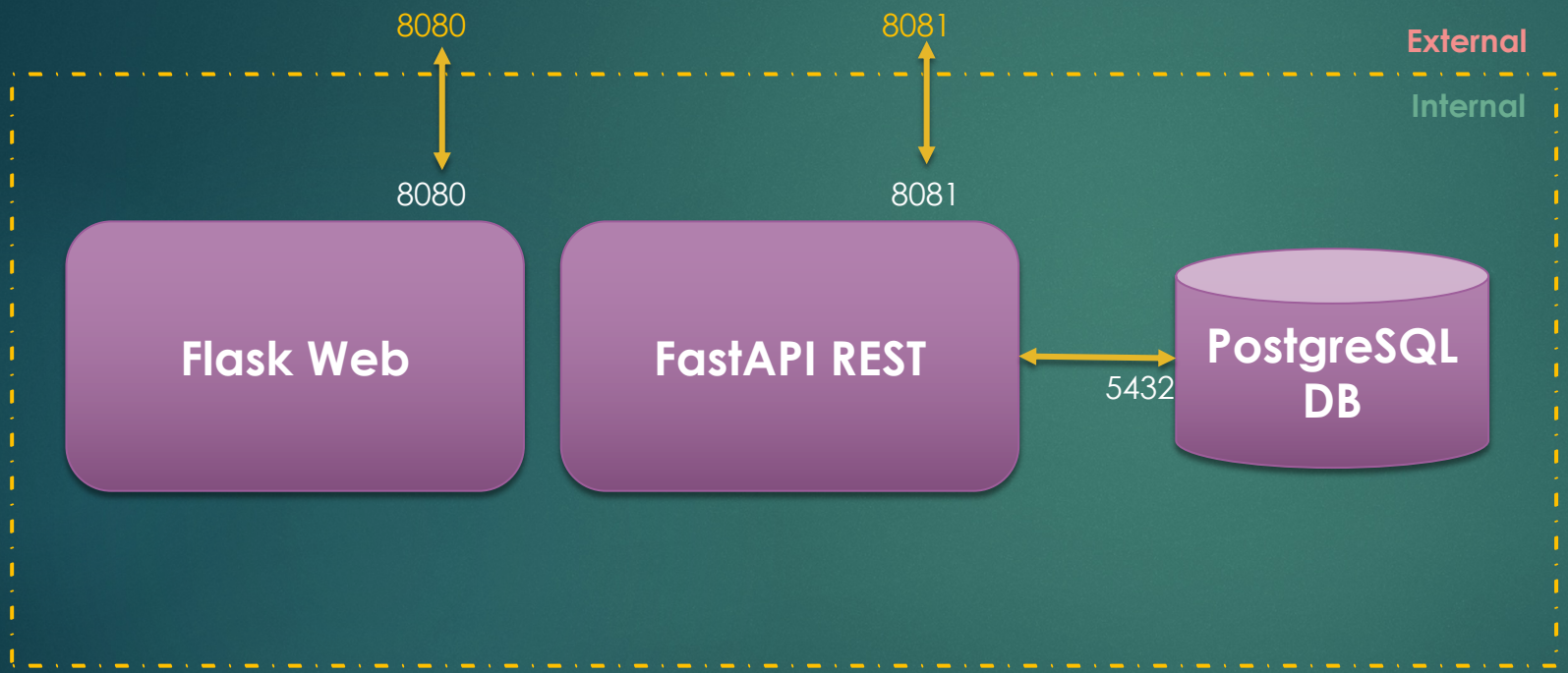
Learn to develop a three-tiered application using Dev Containers

System View



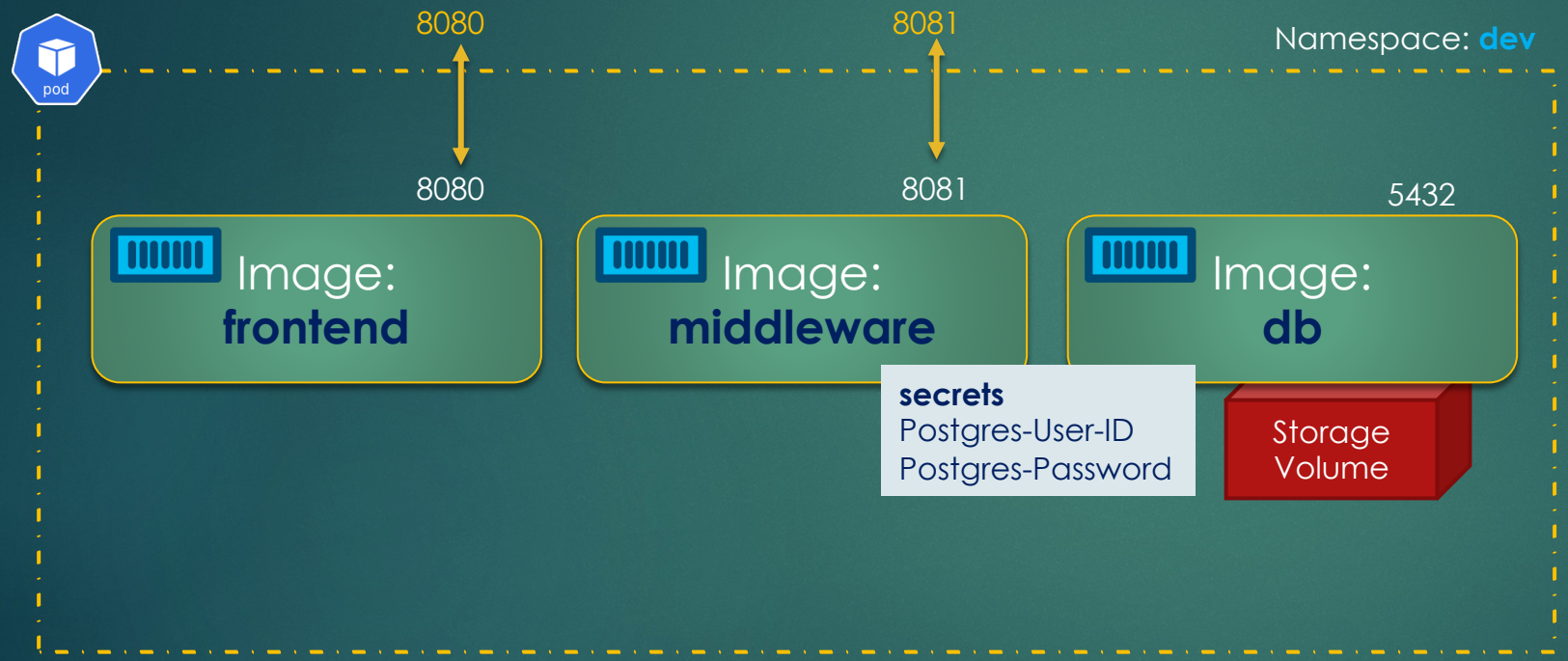
User loads a web site driven by Flask Web (on port: 8080) and then invokes API calls to a FastAPI REST middleware (on port: 8081). API calls are backed by a PostgreSQL DB that stores the needed data.

Component View



- Flask Web delivers the front-end
- FastAPI REST delivers the API Tier
- PostgreSQL delivers the storage tier

One Possible Systems Architecture

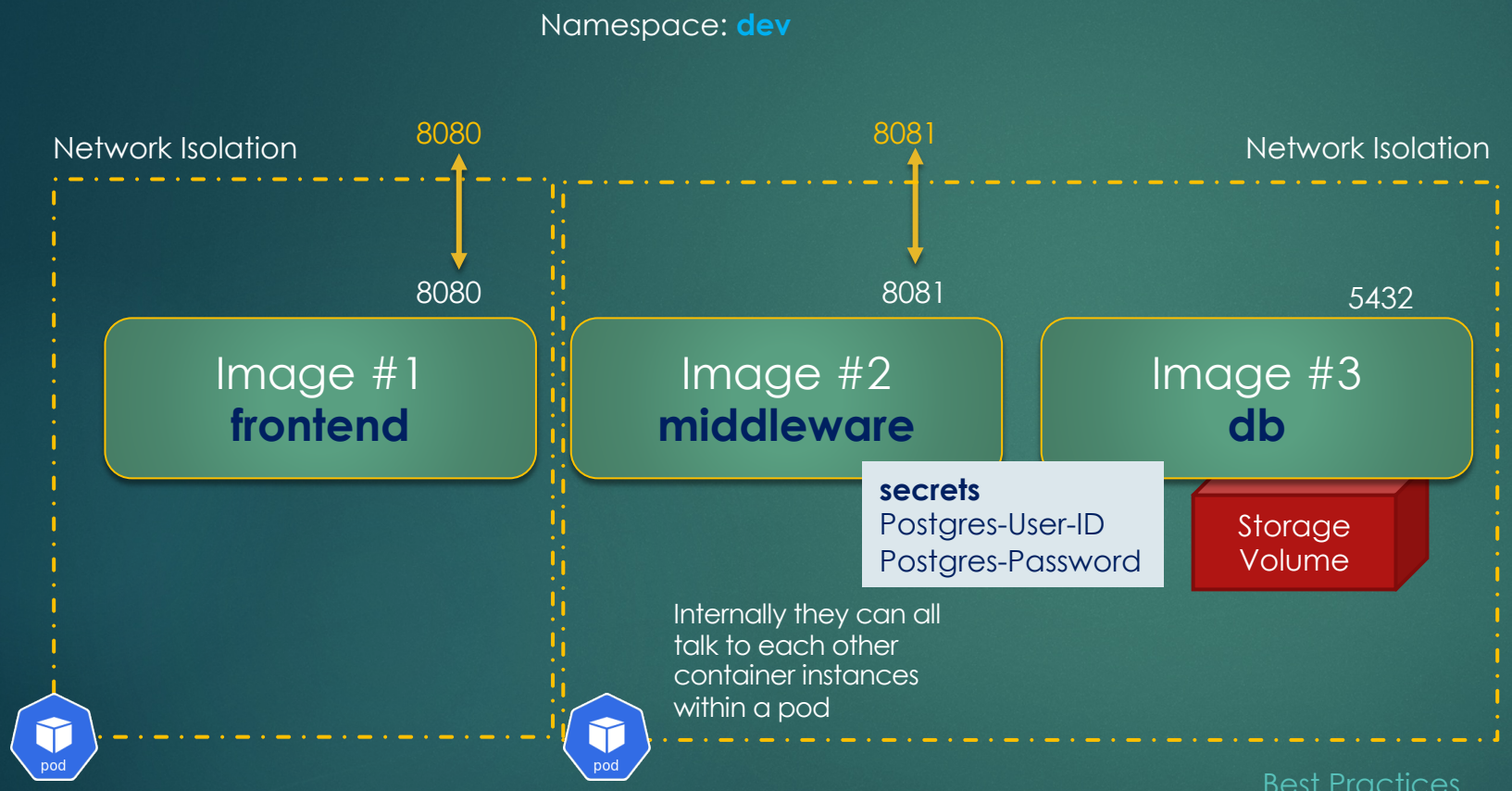


Docker-Images:

1. frontend
2. middleware
3. Postgresql (backed by a storage volume)

Although we can create further network isolation between (middleware + db) from frontend to avoid any possibility of connecting to the db tier from frontend, for our use-case this is adequate.

Alternate Systems Architecture

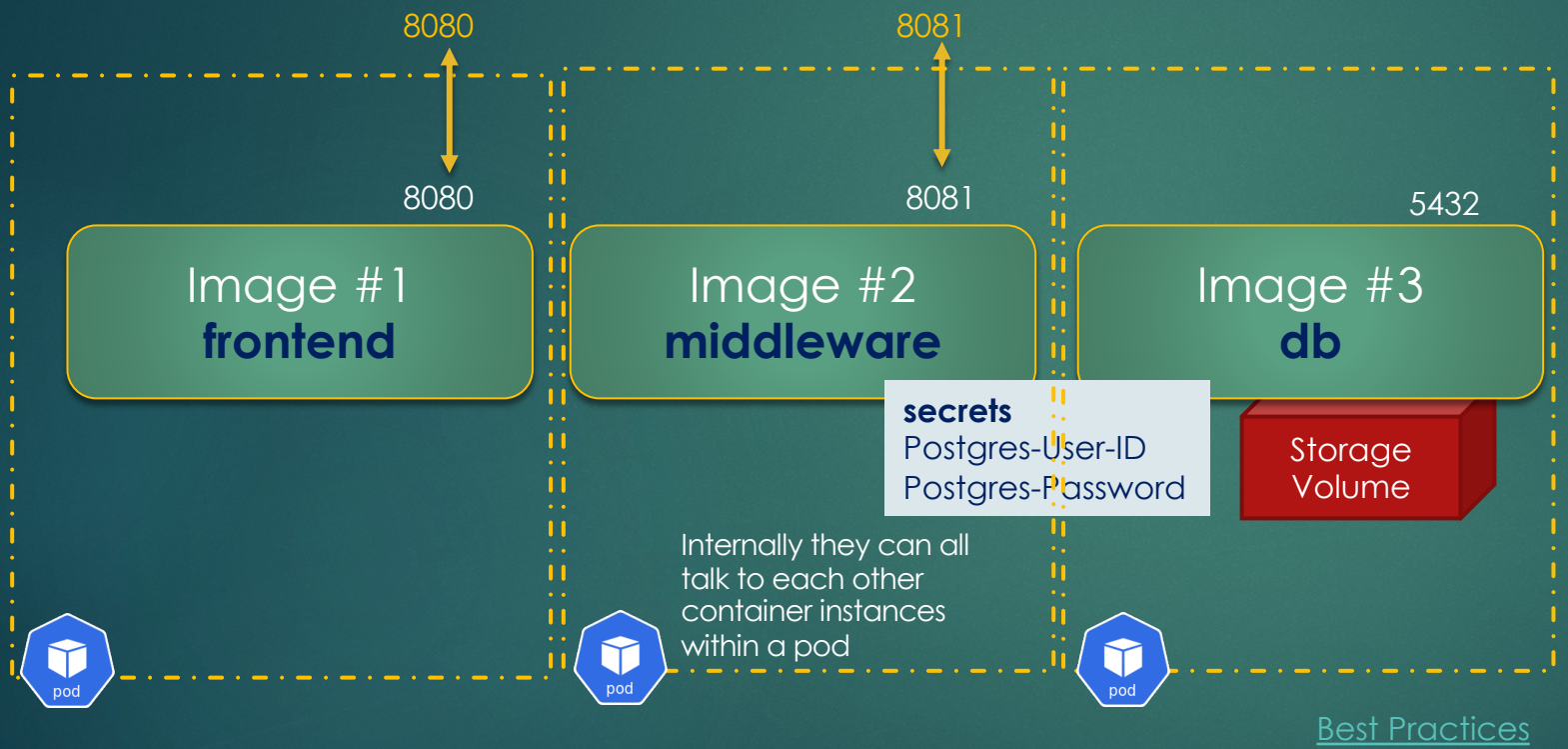


Best Practices

Docker-Images:
1. frontend
2. middleware
3. Postgresql (backed by a storage volume)

Although we can create further network isolation between (middleware + db) from frontend to avoid any possibility of connecting to the db tier from frontend, for our use-case this is adequate.

Alternate Systems Architecture



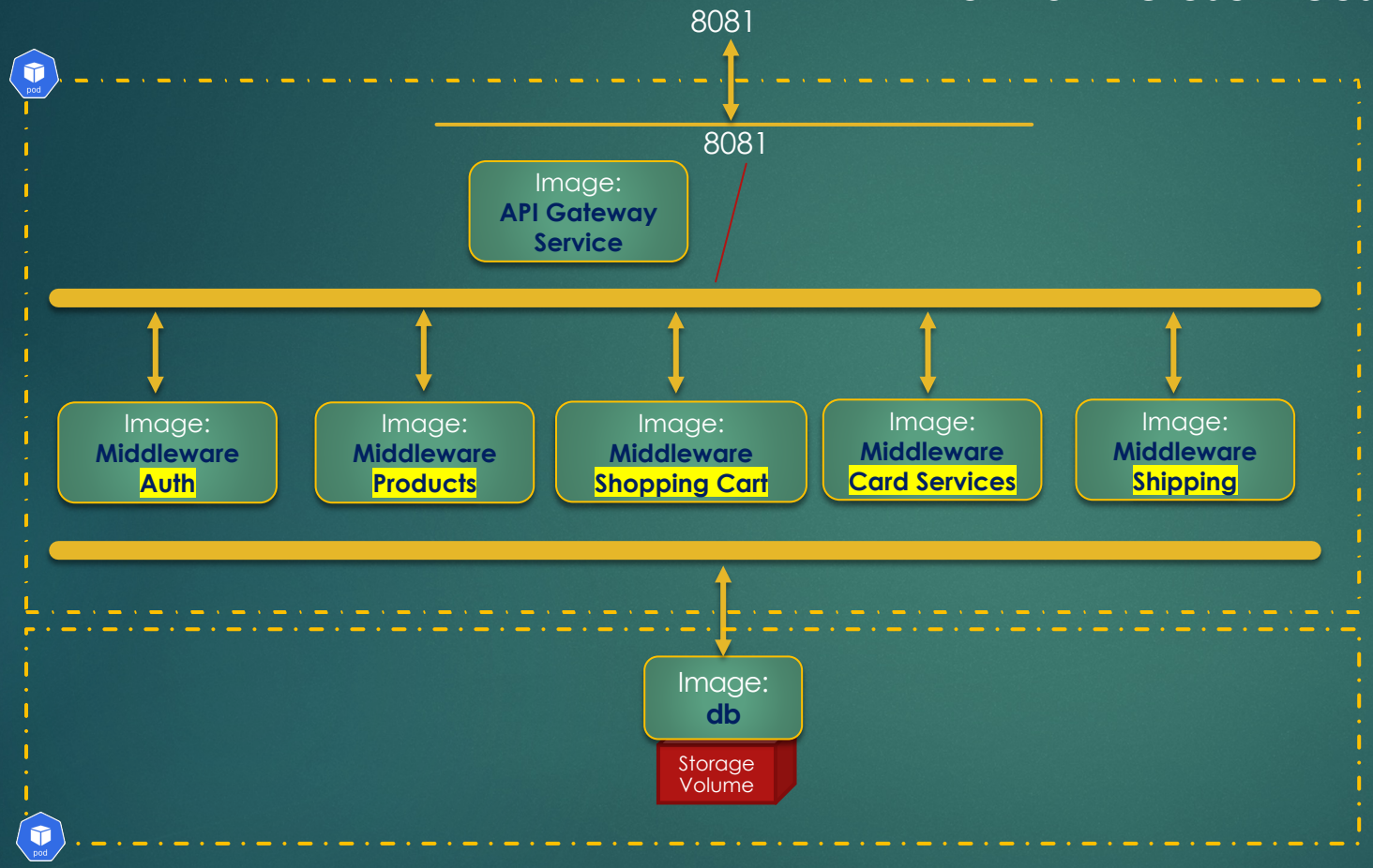
Best Practices

Docker-Images:
1. frontend
2. middleware
3. Postgresql (backed by a storage volume)

Although we can create further network isolation between (middleware + db) from frontend to avoid any possibility of connecting to the db tier from frontend, for our use-case this is adequate.

Yet Another Alternate View

From a Microservices point-of-view



or

Kube-Proxy Ingress Controllers

Quick Links

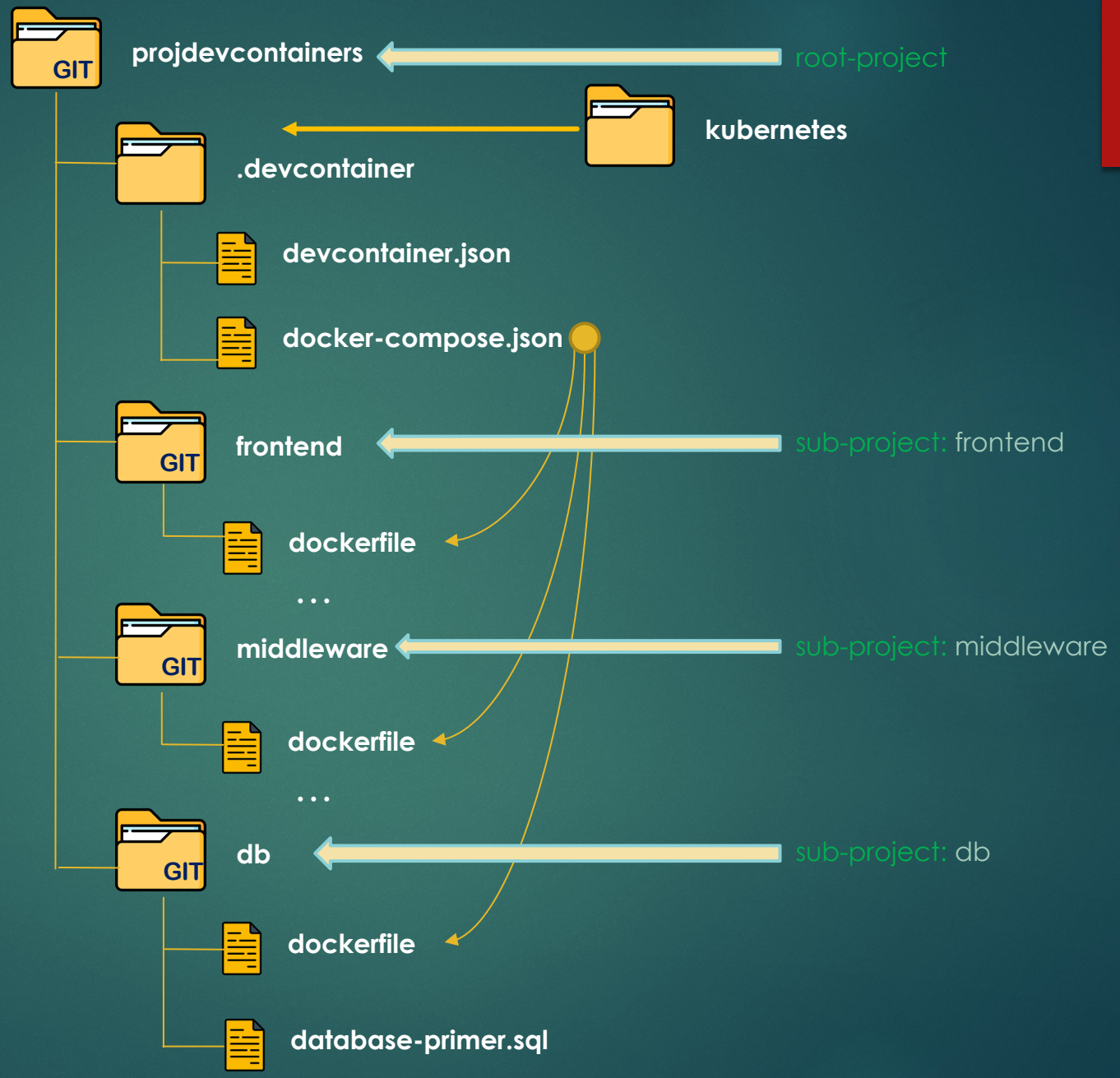
- ✓ Kubernetes Ingress
- ✓ Ingress Controllers

You want to design a pod depending on how you want to scale. You scale at a pod level.

If your desire is to scale the "Auth" container independently with more replicas, then a pod with just that "Auth" container becomes your unit (pod) Whereas, if your objective is to scale all parts (all containers – Auth, Products, Shopping Cart, Card Services, Shipping) uniformly then what you see in the diagram to the left becomes the pod.

NOTE: Any containers in the same pod will share the same resources and local network.

GIT View


GIT
projdevcontainers
root-project
kubernetes
.devcontainer
devcontainer.json
docker-compose.json
GIT
frontend
sub-project: frontend
dockerfile
...
GIT
middleware
sub-project: middleware
dockerfile
...
GIT
db
sub-project: db
dockerfile
database-primer.sql

Container Image Env Diffs

* DEVELOPMENT *

Python Flask Container Image

- ✓ Python
- ✓ Pip
- ✓ Flask
- ✓ Jinja
- ✓ Werkzeug
- ✓ Etc.

dockerfile

Python FastAPI Container Image

- ✓ Python
- ✓ Pip
- ✓ FastAPI
- ✓ Etc.

dockerfile

<- -Interpreted Languages

Golang Container Image

- ✓ Go
- ✓ go-releaser
- ✓ makefile
- ✓ make

dockerfile for DEV

Compiled Languages - - >

dockerfile for non-DEV

Java Container Image

- ✓ JDK
- ✓ Spring Framework
- ✓ maven
- ✓ Other Dependency JARs

dockerfile for DEV

dockerfile for non-DEV

* PRODUCTION *

Python Flask Container Image

- ✓ Python
- ✓ Pip
- ✓ Flask
- ✓ Jinja
- ✓ Werkzeug
- ✓ Etc.

Python FastAPI Container Image

- ✓ Python
- ✓ Pip
- ✓ FastAPI
- ✓ Etc.

Golang Container Image

- ✓ Just the binary with a RUN

Java Container Image

- ✓ App-JAR or WAR
- ✓ JRE

Our Goal

- Create dockerfile for PostgreSQL database
- Create middleware (FastAPI REST) Python code
- Create dockerfile for containerizing middleware
- Create frontend (Flask) Python code
- Create dockerfile for containerizing frontend
- Create a "compose.yaml" (previously docker-compose.yaml) to tie up all the parts into one cluster.
- Use kompose to convert compose.yaml into Kubernetes manifest files.
- Use kubectl (**kind:** Kustomization) to kustomize (combine) into one merged manifest file.
- Use the single manifest file to deploy local K8s cluster.
- Test application

Where does ".devcontainer" fit in?