Cloud:

Modern Apps - Modern Tools

Brian Anstett
Jr. Cloud Developer, 24G

How we run software....

How we write software...

How we run software

Cloud

“Shared pools of configurable computer resources and services that can be rapidly provisioned and accessed over the internet”

ARPANET -> Virtualization -> Saas -> Iaas -> EC2 - Public Cloud

41.5%

29.4%

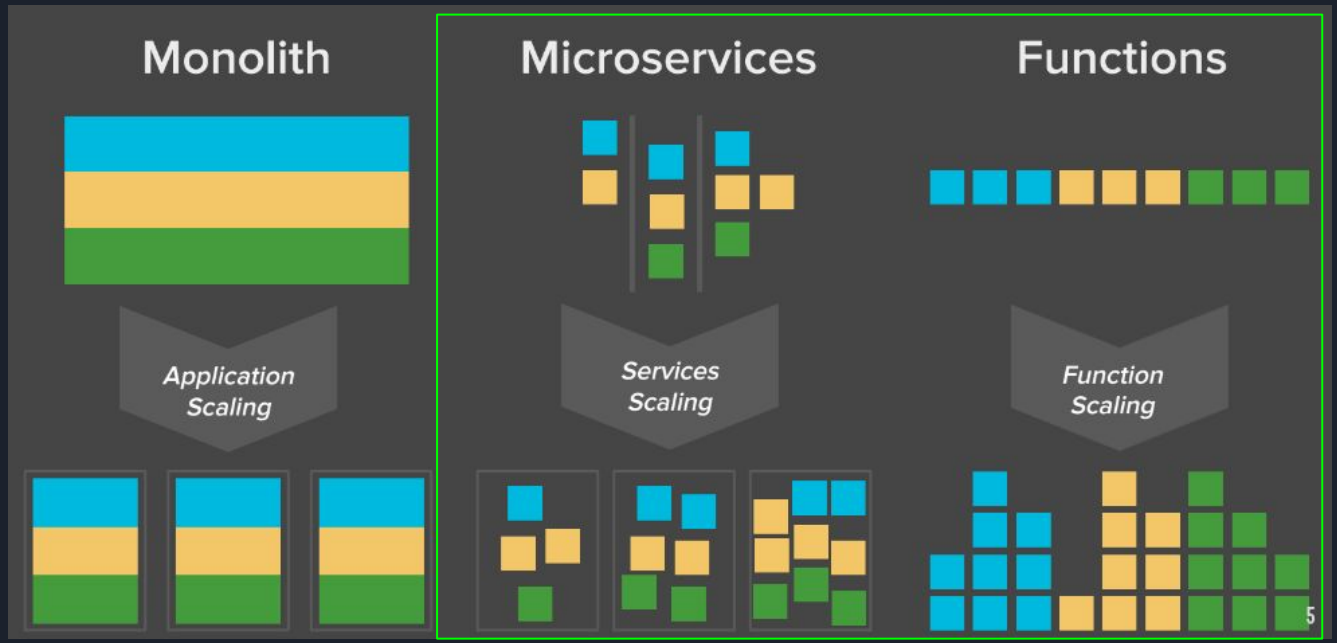
3.0%

- Cost
- Security
- Scalability/ High availability
- Access to advanced technologies (ML)

https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/

How we write software

Monolithic to Microservices



- Scale what needs to be scaled
- Polyglot applications/ developer freedom (kinda)
- Easier, faster, safer deployment

Why does this affect me?

Most wanted languages

Most wanted platforms

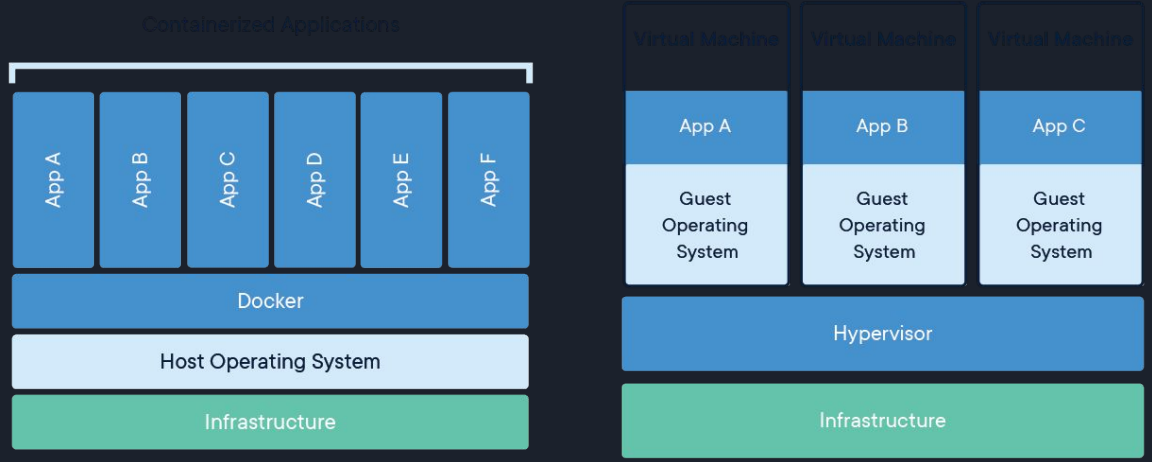
- 83% of enterprise workloads will be in the cloud by 2020
- 85% of Enterprises keep sensitive data on the cloud
- 77% of enterprises have at least one application in the cloud
- 50% of IT spending will be cloud-based in

- Kubernetes: 173%
- Cloud Engineer: 27%
- DevOps: 24%

https://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2018-state-cloud-survey

Docker

- Cgroups
- NameSpaces
- Overlay Filesystem



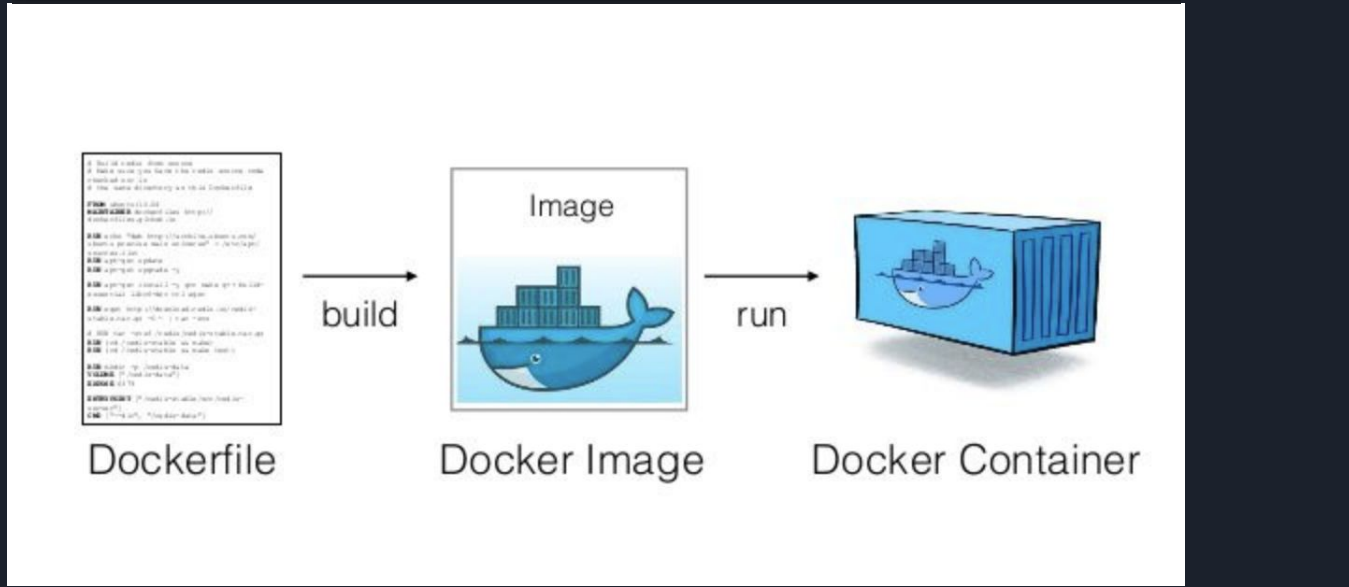
http://man7.org/linux/man-pages/man7/cgroups.7.html
http://man7.org/linux/man-pages/man7/namespaces.7.html
https://wiki.archlinux.org/index.php/Overlay_filesystem

Dockerfile

Section: 1_FirstContainer

Think of a *Dockerfile* as like a class in OOP

```
// docker build -t <name of the image>:<optional tag> <Dockerfile location>
$ docker build -t myimage:v1 .
```



https://docs.docker.com/engine/reference/commandline/build/

Run a container

Section: 1_FirstContainer

Foreground: -it

- Starts process in a container and attaches the console to the process's STDIN/OUT/ERR
- Allocate a pseudo-tty
- When you exit from that main process, your container will stop

Detached: -d

- Container runs in "the background"

-p: publish/map specific ports
-P: pushlish all exposed ports to the host

```
-p hostPort:containerPort
-p hostPort-Range:containerPort-Range
-p 8080:80 // Host port 8080 maps to port 80 on the container
```

```
~$ docker run -d -P httpd
```

```
PORTS
0.0.0.0:32768->80/tcp
```

https://docs.docker.com/engine/reference/run/

Container Registries

Section: 1_FirstContainer

Stores container images (private or public)

- **Docker push**: Pushes images to a registry
- **Docker pull**: Pulls images from a registry
- **Docker tag**: renames an image/alias

```
// optionalHostname/component1/component2/:tag
// us.gcr.io/j-1794/activation2/:v1
```

```
docker push us.gcr.io/g-1575-internal-projects/myimage:v1
```

```
docker pull us.gcr.io/g-1575-internal-projects/myImage:v1
```

Create GCP container registry

1. GCP > Container Registry
2. Enable API
3. Docker login

https://docs.docker.com/engine/reference/commandline/push/
https://docs.docker.com/engine/reference/commandline/pull/
https://docs.docker.com/engine/reference/commandline/tag/

Mounting Host Hardware

Section: 2_MappingExternalDevices

It is possible to “mount” hardware from the *host machine* into a container

**Presenter Only*

--device: Allows you to specify one or more devices that will be accessible within the container

```
$ docker run --device=/dev/sda --rm -id johnny:v1
```

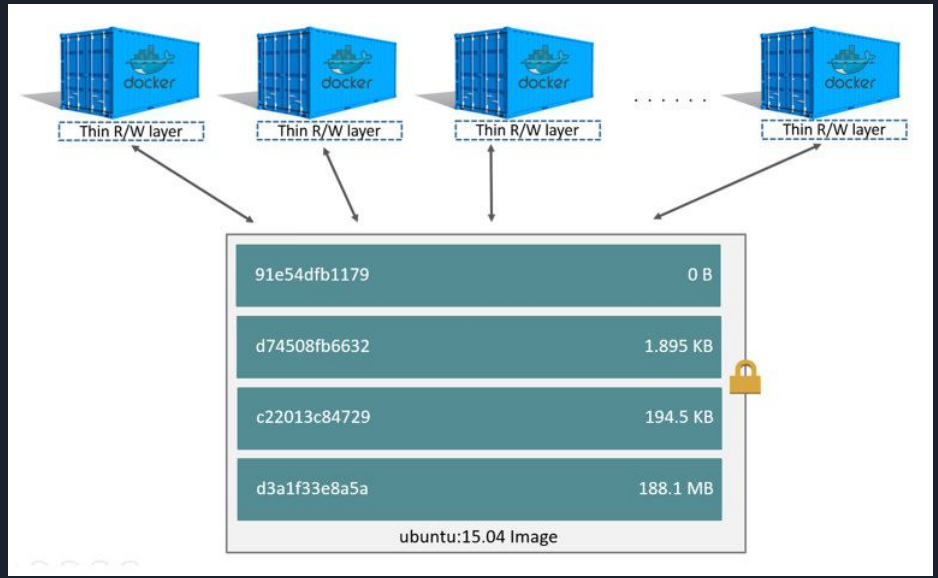
https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities

Container Storage

Section: 3_PersistentStorage

Storage drivers allow us to create data in the writable layer of the container. Files WON'T be persisted after the container is deleted.

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```



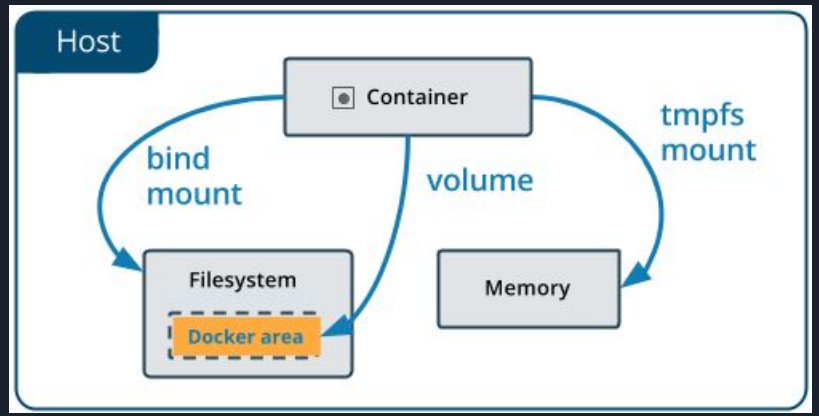
https://docs.docker.com/storage/storagedriver/

Container Storage

Section: 3_PersistentStorage

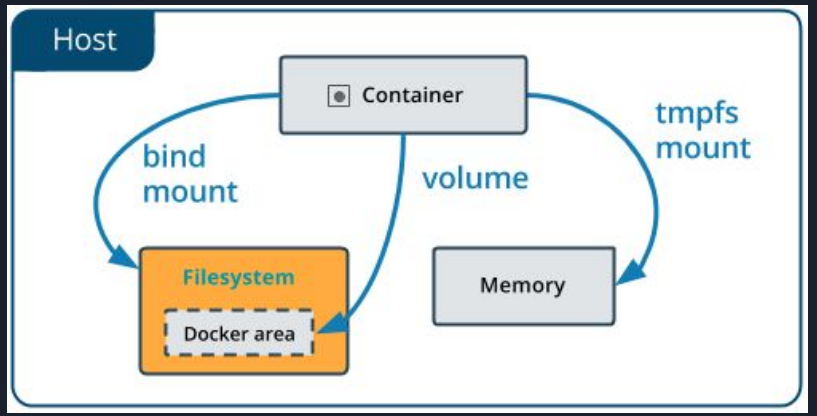
Volumes

- Stored in a part of the host filesystem which is managed by Docker
- Can be mounted into multiple containers
- **Named** vs **anonymous**



Bind Mounts

- Stored anywhere on the host system
- Rely on the host machine's filesystem having specific directory structure



https://docs.docker.com/storage/

Container Storage: Volumes

Section: 3_PersistentStorage

Anonymous Volumes: Created by Docker

```
$ docker run -it --name storagetest --mount type=volume,dst=/tmp/fooBar centos /bin/bash
```

```
FROM ubuntu
RUN mkdir /myvol
RUN echo "hello world" > /myvol/greeting
VOLUME /myvol
```

```
$ docker inspect 23d5f5549cf4 --format '{{.Mounts}}'
[{
  volume 798efd6ec745b337af5c093f874345d02589861079224542d9850497e3a1093c
  /var/lib/docker/volumes/798efd6ec745b337af5c093f874345d02589861079224542d9850497e3a1093c
  /_data /tmp/fooBar
}]
```

Named Volumes: Created by you

https://docs.docker.com/storage/volumes/

Container Storage: Volumes

Section: 3_PersistentStorage

Anonymous Volumes: Created by Docker

Named Volumes: Created by you

```
$ docker volume create foobar
```

```
$ docker run -it --name foobar --mount type=volume,src=foobar,dst=/tmp/foobar centos /bin/bash
```

```
$ docker inspect foobar --format '{{.Mounts}}'
[{
  volume foobar /var/lib/docker/volumes/foobar/_data
  /tmp/foobar
}]
```

https://docs.docker.com/storage/volumes/

Container Storage: Bind Mount

Section: 3_PersistentStorage

Binding Mount

```
hostmachine$ ls /tmp
heyThere.txt
```

```
$ docker run -it --name foobar --mount type=bind,src=/tmp,dst=/tmp centos /bin/bash
[root@4ac5f31690b2 /] ls /tmp
heyThere.txt
```

```
$ docker inspect foobar --format '{{.Mounts}}'
[{
  bind  /tmp /tmp   true rprivate}
}]
```

https://docs.docker.com/storage/bind-mounts/

Container Management

Section: 4_ContainerManagement

docker ps -a[--all]: List containers running (and not running if -a/--all was specified)

```
$docker ps -a
CONTAINER ID        IMAGE     COMMAND                  CREATED             STATUS                      PORTS                     NAMES
d6a568e8ea6a        httpd     "httpd-foreground"       22 hours ago        Up 22 hours                 0.0.0.0:32769->80/tcp
dockerIsCool
e8c9e0736709        centos    "/bin/tail"              4 days ago          Exited (0) 4 days ago
objective_hypatia
208b1b767e6a        centos    "/bin/bash"              4 days ago          Exited (0) 4 days ago
relaxed_mcclintock
```

docker stop/start : Stops/Starts a container

```
$ docker stop <container id or name>
```

docker rm [-v]: Removes a stopped container from your machine. Removes and volumes associated with the container if -v was specified

```
$ docker rm -v <container id or name>
```

Container Management

Section: 4_ContainerManagement

docker images: List images on your machine

```
$ docker images
REPOSITORY                                  TAG                 IMAGE ID            CREATED             SIZE
us.gcr.io/g-1575-internal-projects/node     10_centos           7bd8bff0f108        12 days ago         517MB
httpd                                       latest              2a51bb06dc8b        4 weeks ago         132MB
```

docker rmi: removes one or more images

```
$ docker rmi <images id or name>
```

Container Management

Section: 4_ContainerManagement

docker logs: Shows the *STDOUT* of a container

```
$ docker logs d6a568e8ea6a
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive
globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive
globally to suppress this message
[Sun Dec 16 03:04:31.727436 2018] [mpm_event:notice] [pid 1:tid 140450030589120] AH00489: Apache/2.4.37 (Unix) configured -- resuming
normal operations
[Sun Dec 16 03:04:31.727511 2018] [core:notice] [pid 1:tid 140450030589120] AH00094: Command line: 'httpd -D FOREGROUND'
```

Docker attach: attaches your *STDIN* and *STDOUT* to a running container

```
$ docker attach <container ID or name>
```

Docker exec [-i,t,d]: Run a command in a running container

```
$ docker exec -it <container id or name> /bin/bash
// Inside container
[root]# tail /etc/hosts
```

Containers: What works, what doesn't

Works

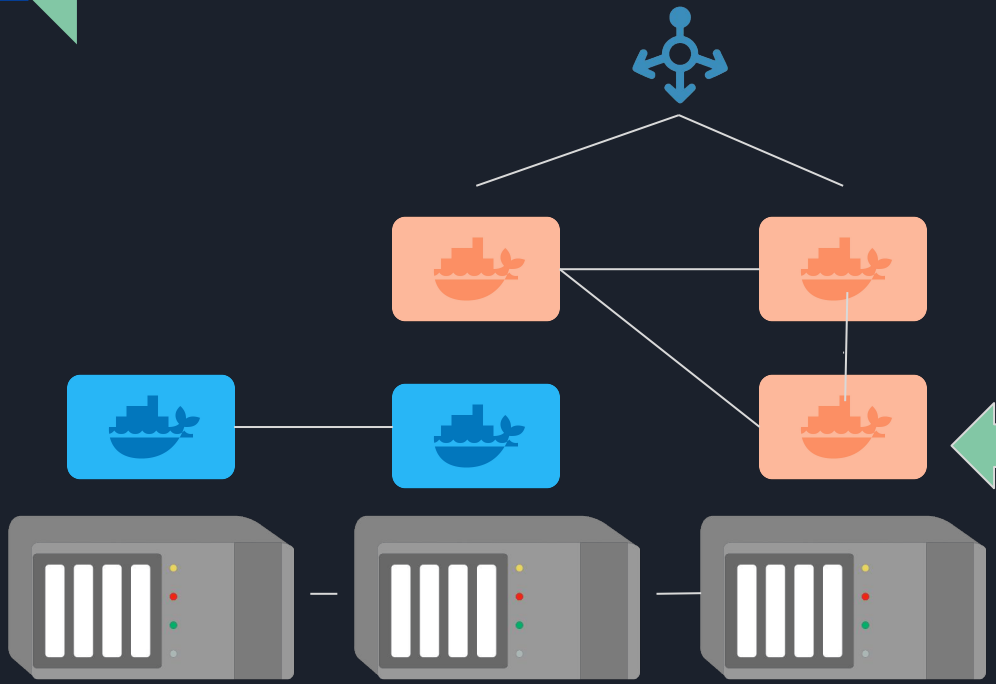
- Package source code, dependencies, and runtime into a shippable unit
- No more "idk, it worked on my computer"
- Widely adopted. Lots of support
- Developer ease

Doesn't work

- Container lifecycle management
- Container networking
- Multiple machines running containers
- Configuration and Secrets

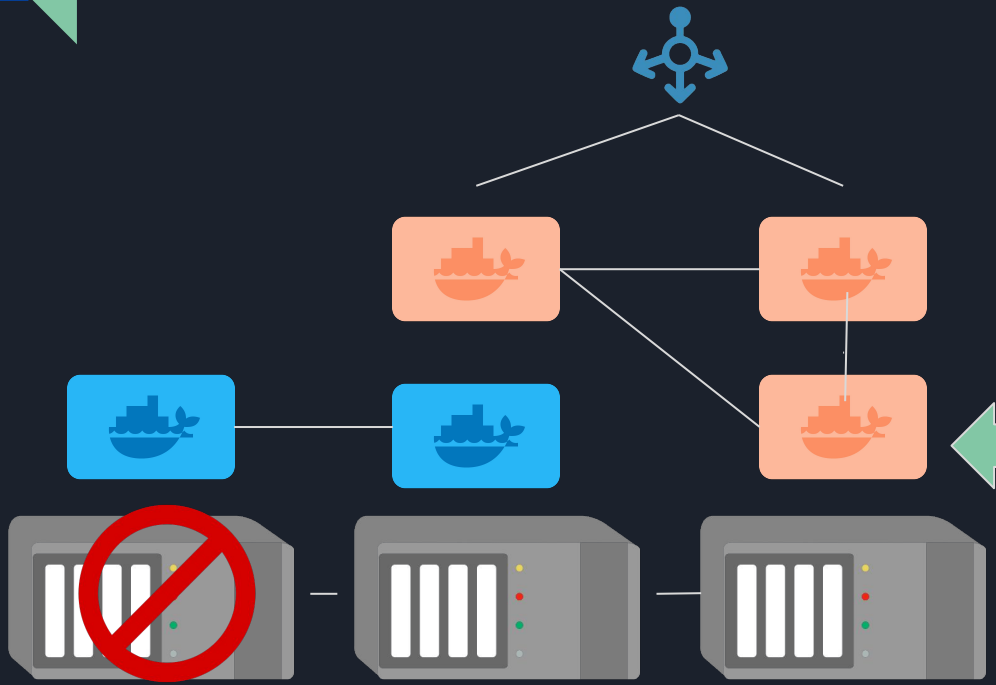
Kubernetes fixes this!

Kubernetes



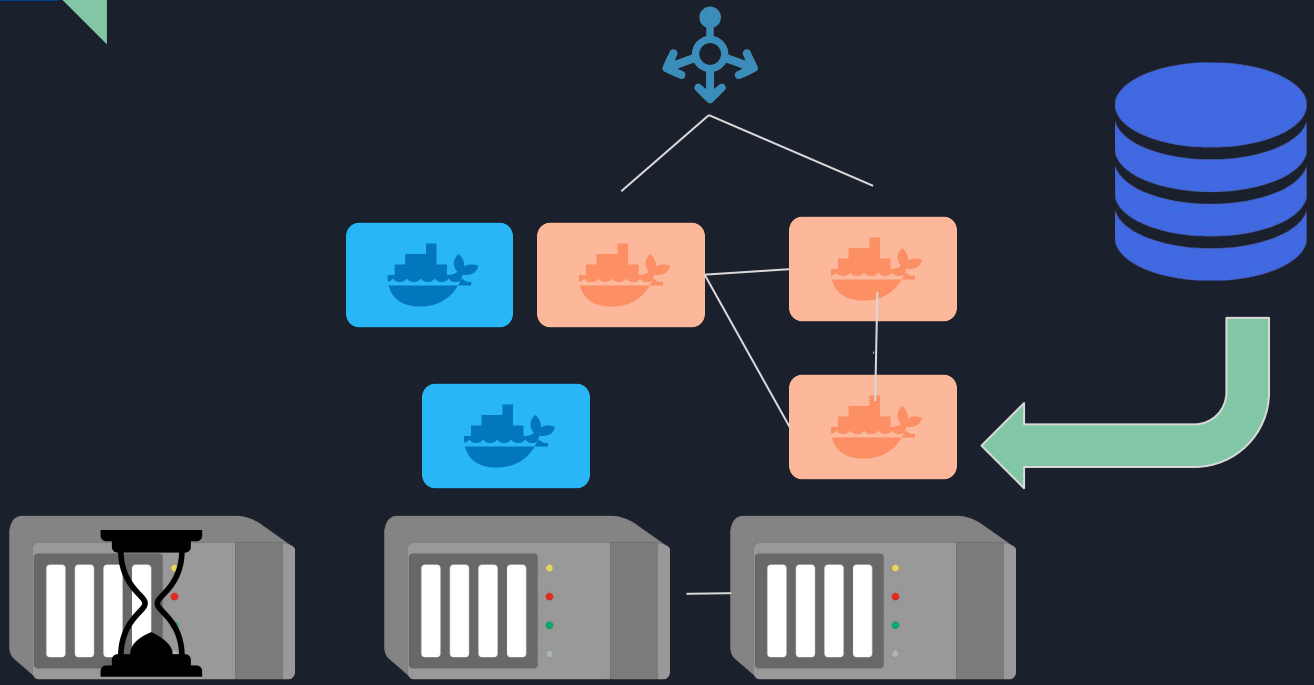
```
Namespace red
    deployment:
        image: redis
        replicas 3
    service:
        type: loadBalancer
        port:6379
Namespace blue
    deployment:
        image: nodejs
        replicas 2
    service:
        type: clusterIp
        port: 80
```

Kubernetes



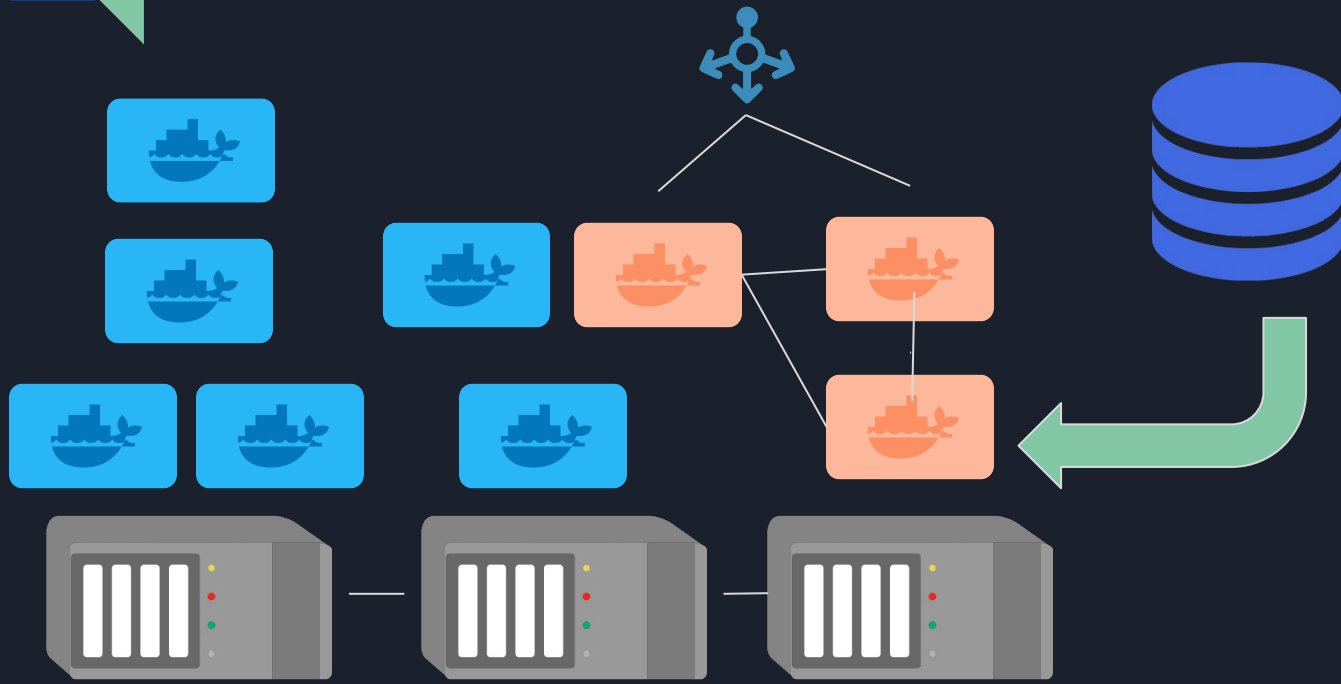
```
Namespace red
    deployment:
        image: redis
        replicas 3
    service:
        type: loadBalancer
        port:6379
Namespace blue
    deployment:
        image: nodejs
        replicas 2
    service:
        type: clusterIp
        port: 80
```

Kubernetes



```
Namespace red
    deployment:
        image: redis
        replicas 3
    service:
        type: loadBalancer
        port:6379
Namespace blue
    deployment:
        image: nodejs
        replicas 2
    service:
        type: clusterIp
        port: 80
```

Kubernetes



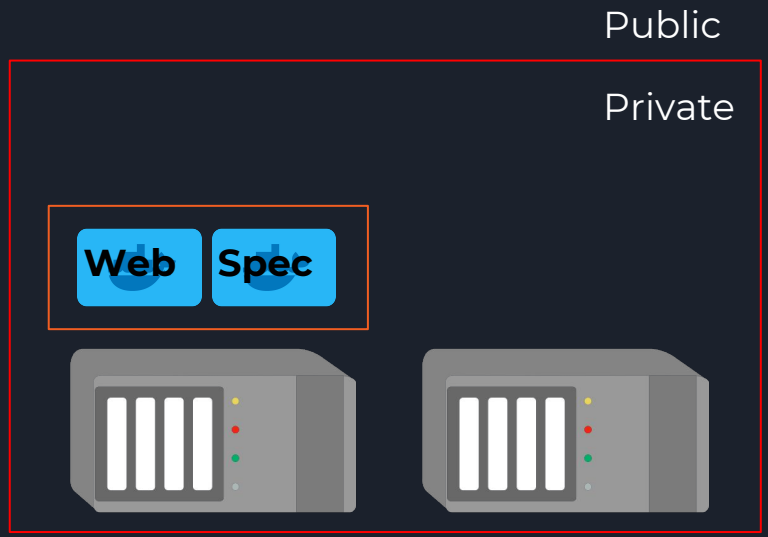
```
Namespace red
    deployment:
        image: redis
        replicas 3
    service:
        type: loadBalancer
        port:6379
Namespace blue
    deployment:
        image: nodejs
        replicas 2
    service:
        type: clusterIp
        port: 80
```

Pods

Section: 1_Pods

Pods: Smallest unit of compute in k8. Encapsulates one or more containers. All container within a pod share storage, network , and lifecycle.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myfirstpod
  labels:
    name: myfirstpod
    environment: development
spec:
  containers:
  - name: webserver
    image: httpd
    ports:
      - containerPort: 80
  - name: spectator
    image: centos
    command: ['tail']
    args: ['-f', '/etc/hosts']
```



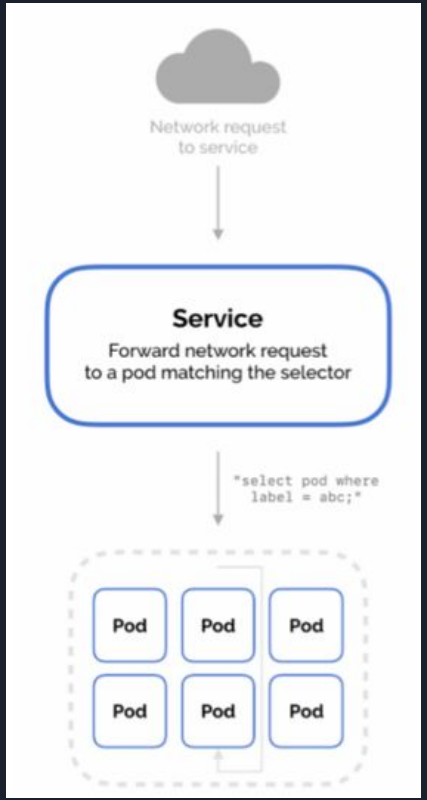
Services

Section: 2_Services

Pods are mortal (and meant to be). “They are born and when they die, they are not resurrected”

The IP addresses of pods can changes at any time so we need a reliable way to forward traffic to pods without the need of IP addresses. A **Service** is exposed as either a…

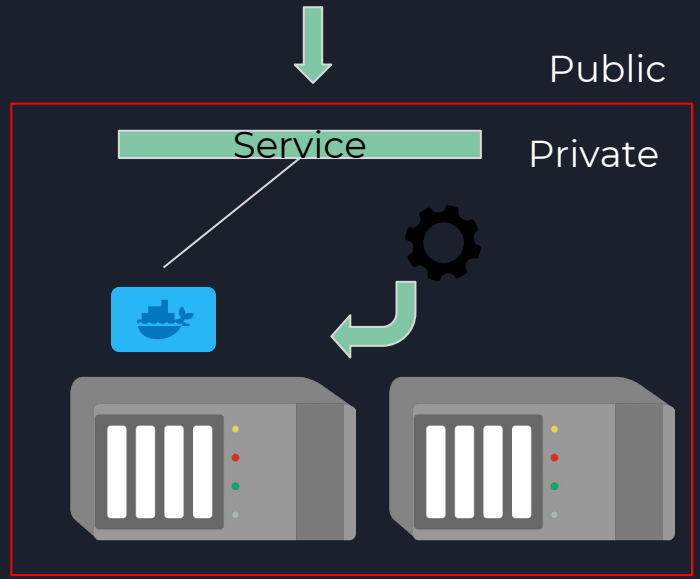
1. **ClusterIP:** Exposes the service on a cluster-internal IP
2. **NodePort**: Exposes the service on each Node’s IP at a static port
    a. (Open firewall)
3. **Load Balancer**: Exposes the service externally using a cloud provider’s LB



Deployments

Section: 3_Deployments

Deployments maintain a desired state of your pods (image, number of replicas, etc.)



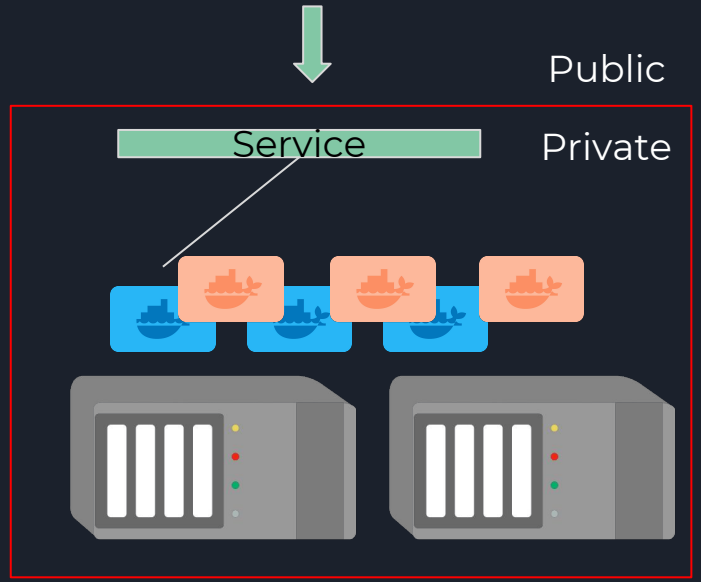

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  labels:
    environmnet: development
    name: httpd-deployment
  name: httpd-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      name: httpd-pod
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
    type: RollingUpdate
  template:
    metadata:
      name: httpd-pod
      labels:
        name: httpd-pod
        environment: development
    spec:
      containers:
      - name: httpd-container
        image: httpd
        ports:
          - containerPort: 80
```

Deployments: Edit state

Section: 3_Deployments

What if we want...

- Our pods to run a new image?
  - Rolling updates
- Add/subtract number of replicas?
  - Manually scale
- Change metadata?

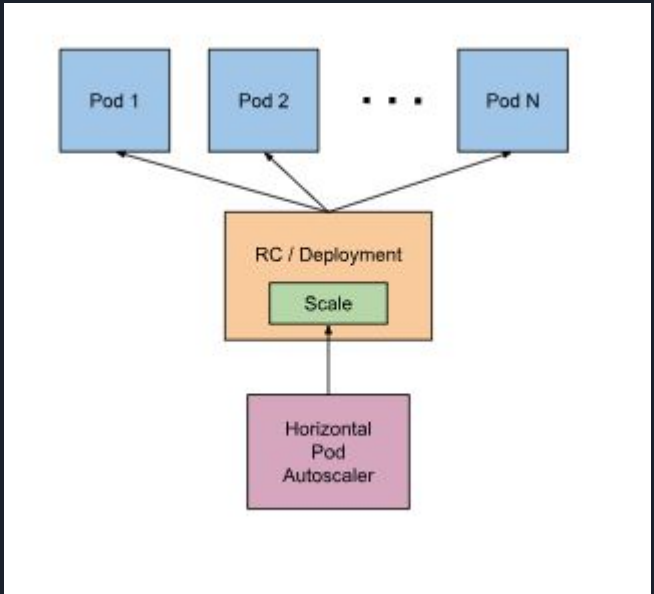


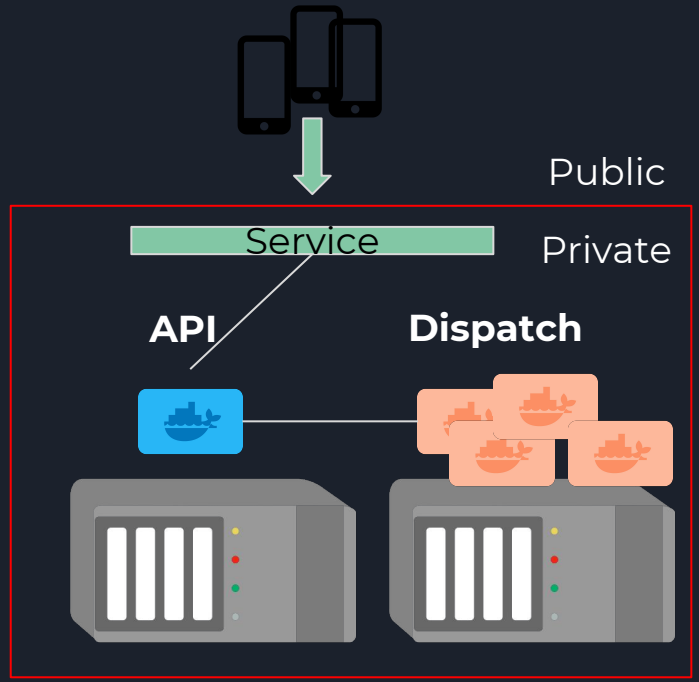
Auto Scaling

Section: 4_AutoScaling

Kubernetes provides a native way to scale your deployment based on resource consumption.

Horizontal Pod Autoscaler





Serverless

Section: reverseName

- No servers to manage. Just the code.
- Triggered by various events