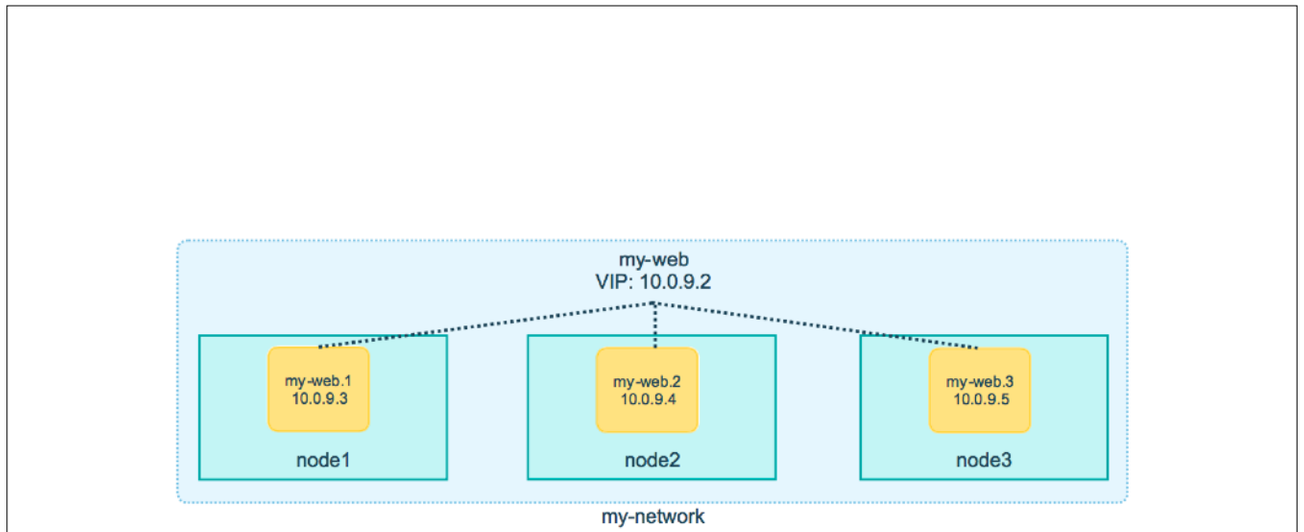
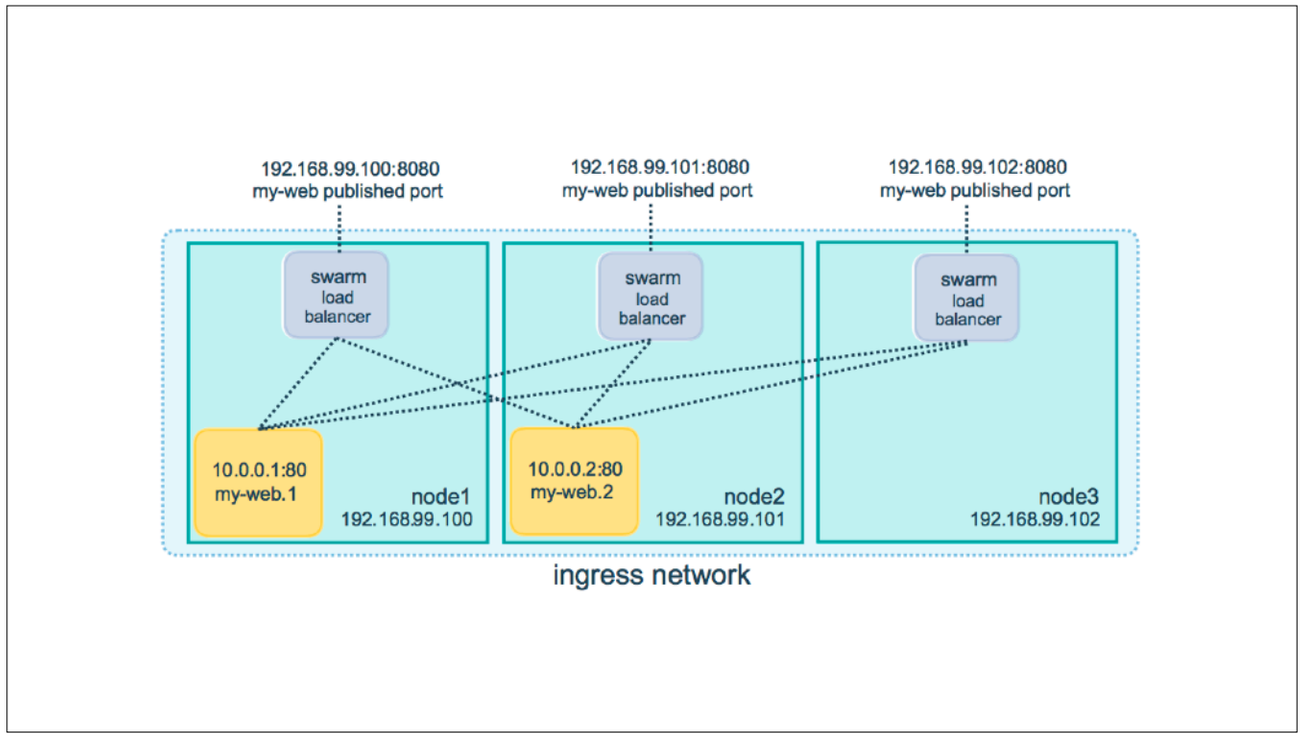
Overlay Multi-Host Networking

- Just choose `--driver overlay` when creating network
- For container-to-container traffic inside a single Swarm
- Optional IPSec (AES) encryption on network creation
- Each service can be connected to multiple networks
  - (e.g. front-end, back-end)

Routing Mesh

- Routes ingress (incoming) packets for a Service to proper Task
- Spans all nodes in Swarm
- Uses IPVS from Linux Kernel
- Load balances Swarm Services across their Tasks
- Two ways this works:
- Container-to-container in a Overlay network (uses VIP)
- External traffic incoming to published ports (all nodes listen)


my-web
VIP: 10.0.9.2
my-web.1
10.0.9.3
node1
my-web.2
10.0.9.4
node2
my-web.3
10.0.9.5
node3
my-network

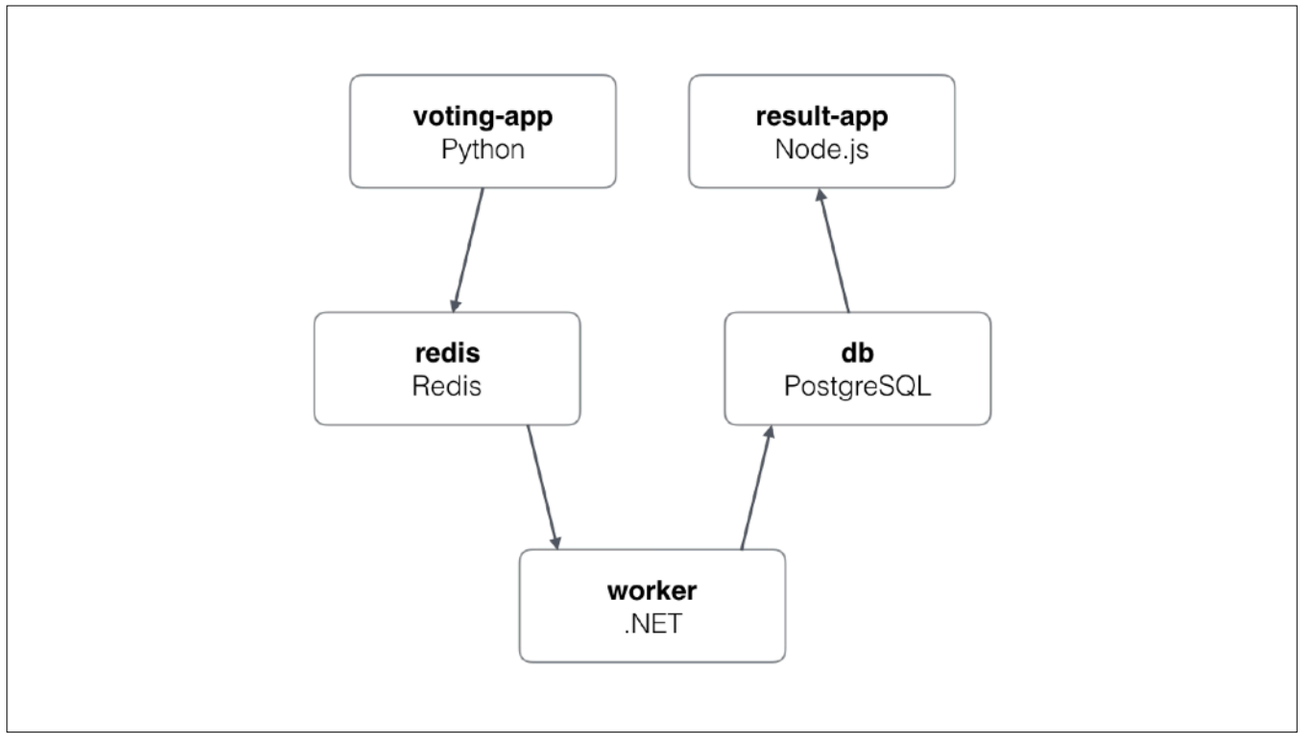

192.168.99.100:8080
my-web published port
192.168.99.101:8080
my-web published port
192.168.99.102:8080
my-web published port
swarm
load
balancer
swarm
load
balancer
swarm
load
balancer
10.0.0.1:80
my-web.1
node1
192.168.99.100
10.0.0.2:80
my-web.2
node2
192.168.99.101
node3
192.168.99.102
ingress network

Routing Mesh Cont.

- This is stateless load balancing
- This LB is at OSI Layer 3 (TCP), not Layer 4 (DNS)
- Both limitation can be overcome with:
- Nginx or HAProxy LB proxy, or:
- Docker Enterprise Edition, which comes with built-in L4 web proxy

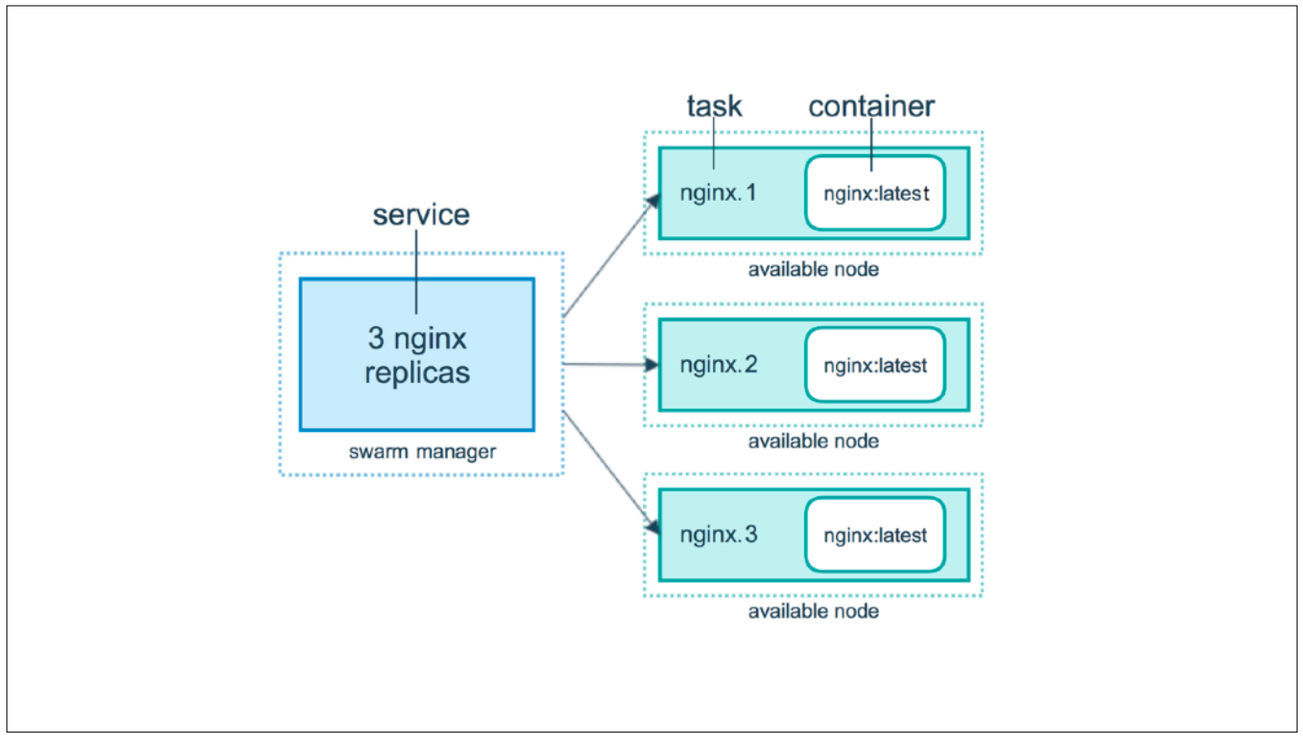
Assignment: Create Multi-Service App

- Using Docker's Distributed Voting App
- use `swarm-app-1` directory in our course repo for requirements
- 1 volume, 2 networks, and 5 services needed
- Create the commands needed, spin up services, and test app
- Everything is using Docker Hub images, so no data needed on Swarm
- Like many computer things, this is ½ art form and ½ science

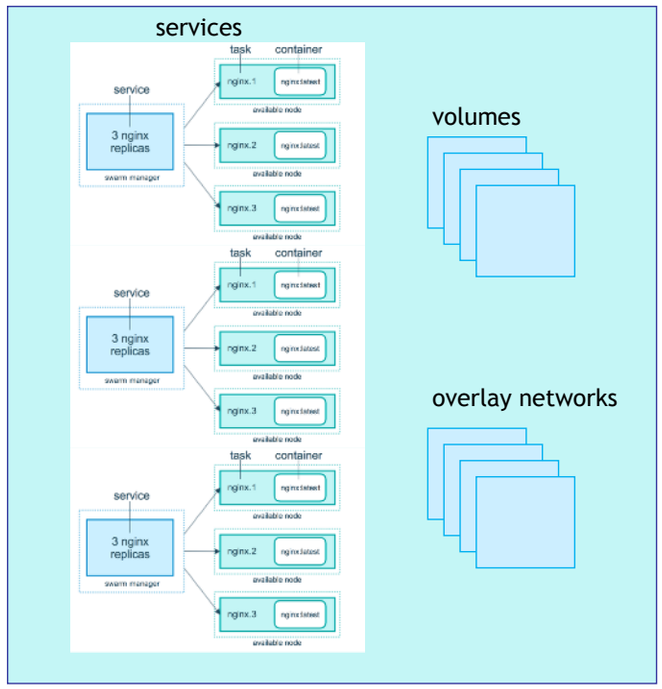

voting-app
Python
result-app
Node.js
redis
Redis
db
PostgreSQL
worker
.NET

Stacks: Production Grade Compose

- In 1.13 Docker adds a new layer of abstraction to Swarm called Stacks
- Stacks accept Compose files as their declarative definition for services, networks, and volumes
- We use `docker stack deploy` rather then docker service create
- Stacks manages all those objects for us, including overlay network per stack. Adds stack name to start of their name
- New `deploy:` key in Compose file. Can't do `build:`
- Compose now ignores `deploy:`, Swarm ignores `build:`
- `docker-compose` cli not needed on Swarm server


task
container
service
nginx.1
nginx:latest
3 nginx
replicas
available node
nginx.2
nginx:latest
swarm manager
available node
nginx.3
nginx:latest
available node

stack!


services
task
container
service
3 nginx replicas
swarm manager
nginx.1
nginx.2
nginx.3
nginx:latest
available node
volumes
overlay networks

Secrets Storage

- Easiest "secure" solution for storing secrets in Swarm
- What is a Secret?
  - Usernames and passwords
  - TLS certificates and keys
  - SSH keys
  - Any data you would prefer not be "on front page of news"
- Supports generic strings or binary content up to 500Kb in size
- Doesn't require apps to be rewritten

Secrets Storage Cont.

- As of Docker 1.13.0 Swarm Raft DB is encrypted on disk
- Only stored on disk on Manager nodes
- Default is Managers and Workers "control plane" is TLS + Mutual Auth
- Secrets are first stored in Swarm, then assigned to a Service(s)
- Only containers in assigned Service(s) can see them
- They look like files in container but are actually in-memory fs
- `/run/secrets/<secret_name>` or `/run/secrets/<secret_alias>`
- Local docker-compose can use file-based secrets, but not secure

Assignment: Create Stack w/ Secrets

- Let's use our Drupal compose file from last assignment
  - (`compose-assignment-2`)
- Rename image back to official `drupal:8.2`
- Remove `build:`
- Add secret via `external:`
- use environment variable `POSTGRES_PASSWORD_FILE`
- Add secret via cli `echo "<pw>" | docker secret create psql-pw -`
- Copy compose into a new yml file on you Swarm node1