1 **S6: Compose**

2 **Docker Compose**
- Why: configure relationships between containers
- Why: save our docker container run settings in easy-to-read file
- Why: create one-liner developer environment startups
- Comprised of 2 separate but related things
- 1. YAML-formatted file that describes our solution options for:
  - containers
  - networks
  - volumes
- 2. A CLI tool docker-compose used for local dev/test automation with those YAML files

3 **docker-compose.yml**
- Compose YAML format has it's own versions: 1, 2, 2.1, 3, 3.1
- YAML file can be used with docker-compose command for local docker automation or..
- With docker directly in production with Swarm (as of v1.13)
- docker-compose --help
- docker-compose.yml is default filename, but any can be used with docker-compose -f

4 **docker-compose CLI**
- CLI tool comes with Docker for Windows/Mac, but separate download for Linux
- Not a production-grade tool but ideal for local development and test
- Two most common commands are
  - docker-compose up   # setup volumes/networks and start all containers
  - docker-compose down # stop all containers and remove cont/vol/net
- If all your projects had a Dockerfile and docker-compose.yml then "new developer onboarding" would be:
  - git clone github.com/some/software
  - docker-compose up

5 **Assignment: Writing A Compose File**
- Build a basic compose file for a Drupal content management system

website. Docker Hub is your friend
- Use the drupal image along with the postgres image
- Use ports to expose Drupal on 8080 so you can localhost:8080
- Be sure to set POSTGRES_PASSWORD for postgres
- Walk though Drupal setup via browser
- Tip: Drupal assumes DB is localhost, but it's service name
- Extra Credit: Use volumes to store Drupal unique data

6 **Using Compose to Build**
- Compose can also build your custom images
- Will build them with docker-compose up if not found in cache
- Also rebuild with docker-compose build
  - or all in one: docker-compose up --build
- Great for complex builds that have lots of vars or build args

7 **Assignment: Build and Run Compose**
- "Building custom drupal image for local testing"
- Compose isn't just for developers. Testing apps is easy/fun!
- Maybe your learning Drupal admin, or are a software tester
- Start with Compose file from previous assignment
- Make your Dockerfile and docker-compose.yml in dir compose-assignment-2
- Use the drupal image along with the postgres image as before
- Use README.md in that dir for details

8 **S7: Services and Swarm**

9 **Containers Everywhere = New Problems**
- How do we automate container lifecycle?
- How can we easily scale out/in/up/down?
- How can we ensure our containers are re-created if they fail?
- How can we replace containers without downtime (blue/green deploy)?
- How can we control/track where containers get started?
- How can we create cross-node virtual networks?
- How can we ensure only trusted servers run our containers?
- How can we store secrets, keys, passwords and get them to the right container (and only that container)?

10 **Swarm Mode: Built-In Orchestration**

- Swarm Mode is a clustering solution built inside Docker
- Not related to Swarm "classic" for pre-1.12 versions
- Added in 1.12 (Summer 2016) via SwarmKit toolkit
- Enhanced in 1.13 (January 2017) via Stacks and Secrets
- Not enabled by default, new commands once enabled
  - docker swarm
  - docker node
  - docker service
  - docker stack
  - docker secret

11 **Swarm Graphic 1**

12 **Swarm Graphic 2**

13 **Swarm Graphic 3**

14 **Swarm Graphic 4**

15 **Swarm Logo**

16 **docker swarm init: What Just Happened?**

- Lots of PKI and security automation
  - Root Signing Certificate created for our Swarm
  - Certificate is issued for first Manager node
  - Join tokens are created
- Raft database created to store root CA, configs and secrets
  - Encrypted by default on disk (1.13+)
  - No need for another key/value system to hold orchestration/secrets
  - Replicates logs amongst Managers via mutual TLS in "control plane"

17 **Creating 3-Node Swarm: Host Options**

- A. play-with-docker.com
  - Only needs a browser, but resets after 4 hours
- B. docker-machine + VirtualBox
  - Free and runs locally, but requires a machine with 8GB memory
- C. Digital Ocean + Docker install
  - Most like a production setup, but costs $5-10/node/month while learning
  - Use my referral code in section resources to get $10 free

- D. Roll your own
  - docker-machine can provision machines for Amazon, Azure, DO, Google, etc.
  - Install docker anywhere with get.docker.com

18 **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)

19 **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)

20 **Routing Mesh Graphic 1**

21 **Routing Mesh Graphic 2**

22 **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

23 **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

24 **Example Voting App**

25 **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

26 **Swarm Graphic 3**

27 **Stack Graphic 1**

28 **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

29 **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

30 **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

31 ☐ **Full App Lifecycle With Compose**

- Live The Dream!
- Single set of Compose files for:
- Local docker-compose up development environment
- Remote docker-compose up CI environment
- Remote docker stack deploy production environment
- Note: docker-compose -f a.yml -f b.yml config mostly works
- Note: Compose extends: doesn't work yet in Stacks
- Fast moving part of toolset. Expect this to change/fix.

32 ☐ **Container Registries**

- An image registry needs to be part of your container plan
- More Docker Hub details including auto-build
- How Docker Store (store.docker.com) is different then Hub
- How Docker Cloud (cloud.docker.com) is different then Hub
- Use new Swarms feature in Cloud to connect Mac/Win to Swarm
- Install and use Docker Registry as private image store
- 3rd Party registry options

33 ☐ **Docker Hub: Digging Deeper**

- The most popular public image registry
- It's really Docker Registry plus lightweight image building
- Let's explore more of the features of Docker Hub
- Link GitHub/BitBucket to Hub and auto-build images on commit
- Chain image building together

34 ☐ **Docker Store: What Is It For?**

- Download Docker "Editions"
- Find certified Docker/Swarm plugins and commercial certified images

35 ☐ **Docker Cloud: CI/CD and Server Ops**

- Web based Docker Swarm creation/management
- Uses popular cloud hosters and bring-your-own-server
- Automated image building, testing, and deployment
- More advanced then what Docker Hub does for free
- Includes an image security scanning service

36 **Running Docker Registry**

- A private image registry for your network
- Part of the docker/distribution GitHub repo
- The de facto in private container registries
- Not as full featured as Hub or others, no web UI, basic auth only
- At its core: a web API and storage system, written in Go
- Storage supports local, S3/Azure/Alibaba/Google Cloud, and OpenStack Swift

37 **Running Docker Registry Cont.**

- Look in section resources for links to:
- Secure your Registry with TLS
- Storage cleanup via Garbage Collection
- Enable Hub caching via "--registry-mirror"

38 **Run a Private Docker Registry**

- Run the registry image on default port 5000
- Re-tag an existing image and push it to your new registry
- Remove that image from local cache and pull it from new registry
- Re-create registry using a bind mount and see how it stores data

39 **Registry and Proper TLS**

- "Secure by Default": Docker won't talk to registry without HTTPS
- Except, localhost (127.0.0.0/8)
- For remote self-signed TLS, enable "insecure-registry" in engine

40 **Run a Private Docker Registry Recap**

- Run the registry image
  - docker container run -d -p 5000:5000 --name registry registry
- Re-tag an existing image and push it to your new registry
  - docker tag hello-world 127.0.0.1:5000/hello-world
  - docker push 127.0.0.1:5000/hello-world
- Remove that image from local cache and pull it from new registry
  - docker image remove hello-world

  - •docker image remove 127.0.0.1:5000/hello-world
  - •docker pull 127.0.0.1:5000/hello-world
- Re-create registry using a bind mount and see how it stores data
  - •docker container run -d -p 5000:5000 --name registry -v $(pwd)/registry-data:/var/lib/registry registry

41 **Remember To Cleanup!**

- No containers created in this Lecture are required for future Lectures

42 **Private Docker Registry with Swarm**

- Works the same way as localhost
- Because of Routing Mesh, all nodes can see 127.0.0.1:5000
- Remember to decide how to store images (volume driver)
- NOTE: All nodes must be able to access images
- ProTip: Use a hosted SaaS registry if possible

43 **Container Startup: Shell vs. Exec Form**

- The first process to start in Linux is known as "PID 1"
- Ideally, our containers only run one process
- Depending on subtle differences in startup, we may be running two
- Shell: CMD sleep 10000
- Exec: CMD ["sleep", "10000"]
- Avoid Shell format when you can, which is almost always

44 **Container Startup: ENTRYPOINT/CMD**

- There seems to be two Dockerfile commands for running our app: ENTRYPOINT, and CMD
- At least one of them should be in Dockerfile (might be in FROM img)
- Both can be used, and CMD will compliment ENTRYPOINT, where ENTRYPOINT will be the executable, CMD will be the arguments
- Anything after docker run imagename is overriding CMD
- ENTRYPOINT should be used if container is a cli tool w/ output
- In that case, CMD is default arguments

45 **Container Startup: docker-entrypoint.sh**

- Added script via your Dockerfile
- For when you need to do stuff before your app starts
- Do this over running scripts in CMD (you'll have the PID 1 problem)
- docker-entrypoint.sh uses ENTRYPOINT and shell features to run a script first, before CMD runs your app

- Some Hub images may have a dir that autoruns any script in it

46 **Getting Data In and Out of Docker**

- First, nothing beats Dockerfile for re-producing images
- But sometimes You just need to move data/images around
- We've covered push/pull from registry
- We've covered bind mounts
- Let's cover some others

47 **Getting Data In and Out Cont.**

- docker image save to put image in a tar file
- docker image load to restore image from tar file
- docker container export to put container contents in tar file
- docker image import to put file structure tar in new image
- Use named volume and access host path
-