Section Overview: Updates and Healthchecks

- Day 1 vs. Day 2 Operations
- Rolling Service Updates
- Timeline of a Service Update
- How Healthchecks Affect Updates
- Creating Your Own Healthchecks
- Gracefully Handle Update Issues with Rollbacks

Section Requirements

- Created a 3-node (or more) Swarm with your environment of choice
- Created the Swarm Visualizer service from previous lecture
- Cleared other stacks/services/containers/volumes/networks

Run The Swarm Visualizer?

- Start a service for the Docker Swarm Visualizer
- This is a useful learning graphic that shows us how tasks move around
- https://github.com/dockersamples/docker-swarm-visualizer
- docker service create \
  --name=viz \
  --publish=8080:8080/tcp \
  --constraint=node.role==manager \
  --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  dockersamples/visualizer

Rolling Service Update Basics

- Review Service Updates in earlier section if refresher needed
- Service Update default: replace each replica, one at a time
- We can customize a lot of that update process
- Before going live in production, test this **a lot**
- Different apps handle session and reconnection differently
- Few apps do this well, even with orchestrator help

New Testing Tool: httping

- Test a HTTP(S) connection similar to how ping works
- Shows HTTP response code, colors, CLI GUI and more
- https://hub.docker.com/r/bretfisher/httping/
- We'll use my container image, but also avail on
  - brew install httping
  - apt-get install httping
  - no easy build for Windows
- docker run bretfisher/httping localhost works everywhere!

New Testing App: browncoat

- Simple web app with settings for acting badly

  "I aim to misbehave" -- *Malcom Reynolds, Serenity (Firefly)*
- https://hub.docker.com/r/bretfisher/browncoat/
- By default, no healthcheck, but functions correctly
- Different image tags and envvars for changing behavior
- Purpose: test rolling updates, rollbacks, and healthchecks

Testing Rolling Service Updates

- Run a basic service, then update it while we httping
  > docker network create --driver overlay --attachable verse
  > docker service create --name firefly -p 80:80 --network verse \
  --replicas 5 bretfisher/browncoat:v1
- Lets watch it with http (another window)
  > docker run --rm --network verse bretfisher/httping -i .1 -GsY firefly/healthz
- Lets update it
  > docker service update --image bretfisher/browncoat:v2 firefly

Slow Our Container Startup

- Not too bad, but what if containers take 5 seconds to start up
  > docker service update --env-add DELAY_STARTUP=5000 firefly
- Not good, lets dig in with next Lecture

Lecture Cleanup

- Stop the httping container
  - Windows: ctrl-c and docker stop <container name>
  - Linux/macOS: ctrl-c

- Remove the service and network we created

> docker service rm firefly

> docker network rm verse

Timeline Of A Service Update

- Swarm will upgrade N instances at a time, change with update-parallelism
- New tasks are created, and their desired state is set to **Ready**
  - Ensures resource availability (**Pending**)
  - Pulls the image if necessary (**Preparing**)
  - Creates the container ... without starting it (**Ready**)
- If a task fails to get to **Ready** state, it retries with a new task
- When tasks are **Ready**, it sets the old tasks desired state to **Shutdown**
- When the old tasks are **Shutdown**, it starts the new tasks, set to **Running**
- Then it waits for the update-delay, and continues with the next task batch

Update Options

Batteries included, but swappable

--stop-grace-period	Time to wait before force killing a container (ns|us|ms|s|m|h)
--stop-signal string	Signal to stop the container
--update-delay	Delay between updates
--update-failure-action	Action on update failure ("pause"|"continue"|"rollback")
--update-max-failure-ratio	Failure rate to tolerate during an update
--update-monitor	Duration after each task update to monitor for failure
--update-order	Update order ("start-first"|"stop-first")
--update-parallelism	Maximum number of tasks updated simultaneously

More Timeline Things To Care About

- Healthchecks, if enabled, affect **Running** state
- What if my new version fails starting or healthcheck?
  - --update-max-failure-ratio
  - --update-failure-action ("pause"|"continue"|"rollback")
  - Usually: test = pause or continue to troubleshoot
  - Usually: prod = rollback
- --update-order, default to stop-first ("start-first"|"stop-first")

Service Update Examples

- Monitor for 5min before next, rollback on failure
  - docker service update --update-failure-action rollback --update-monitor 5m node
- Update 5 at a time, up to 25% can fail until failure action
  - docker service update --update-parallelism 5 --update-max-failure-ratio .25
  - For if you have lots of containers and distributed failures are ok
- Start new container first before killing old one
  - docker service update --update-order start-first wordpress
  - Good for single-replica services (you didn't need HA)
  - Not good for databases with volume storage (avoid file multi-access)

Assignment: Trying Update Options

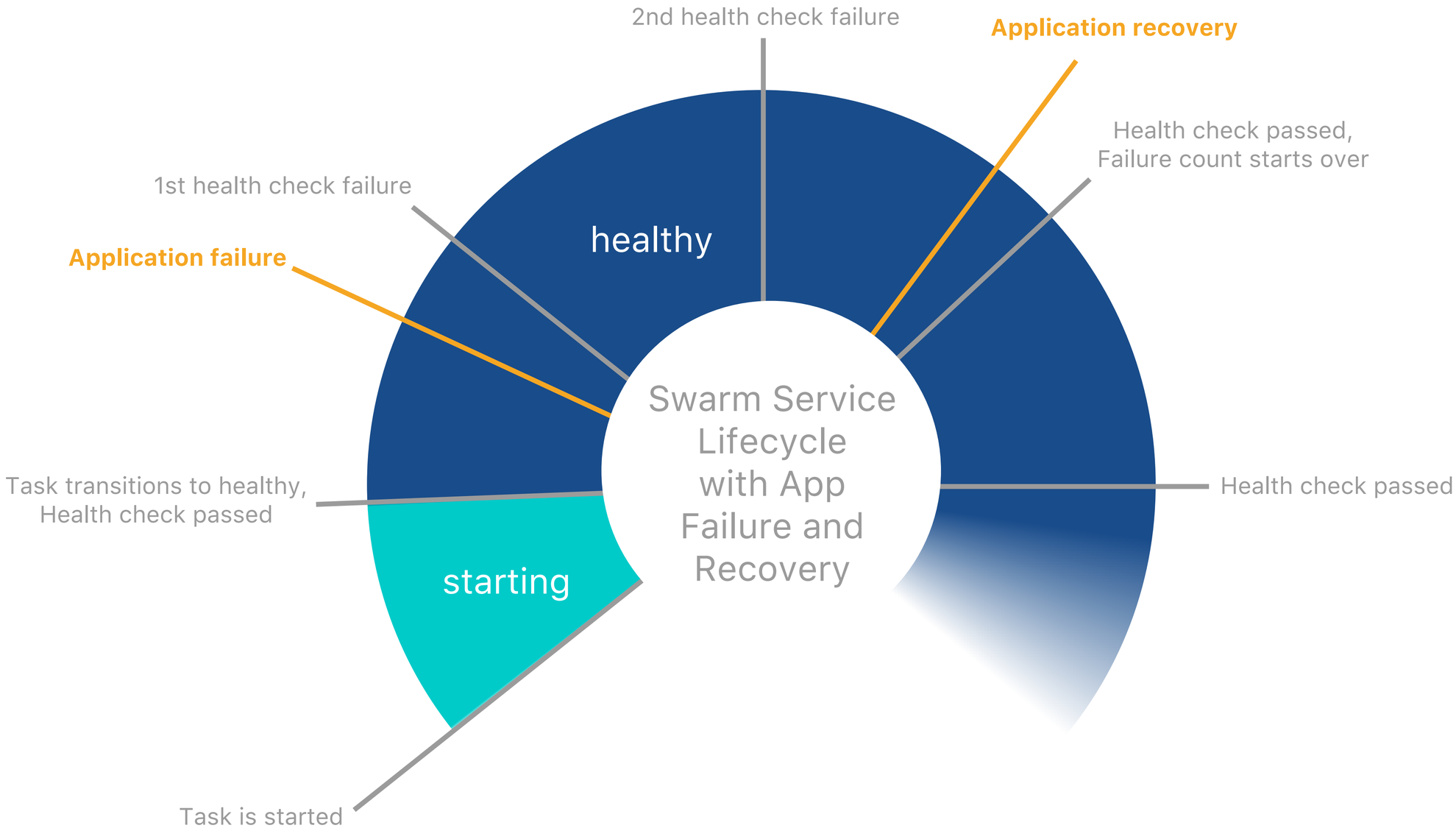
- Create service (constrain to current node)
  - > docker network create --driver overlay --attachable verse
  - > docker service create --name firefly -p 80:80 --network verse --replicas 5 --constraint "node.hostname==node1" bretfisher/browncoat:v1
- Monitor for 15 seconds before next task (no-op)
  - > docker service update --update-monitor 15s firefly
  - > docker service inspect --pretty firefly
- Update 5 at a time, force a update without changes
  - > docker service scale firefly=15
  - > docker service update --update-parallelism 5 --force firefly
- Start new container first before killing old one (watch with docker events)
  - > docker service scale firefly=1
  - > docker service update --update-order start-first --force firefly

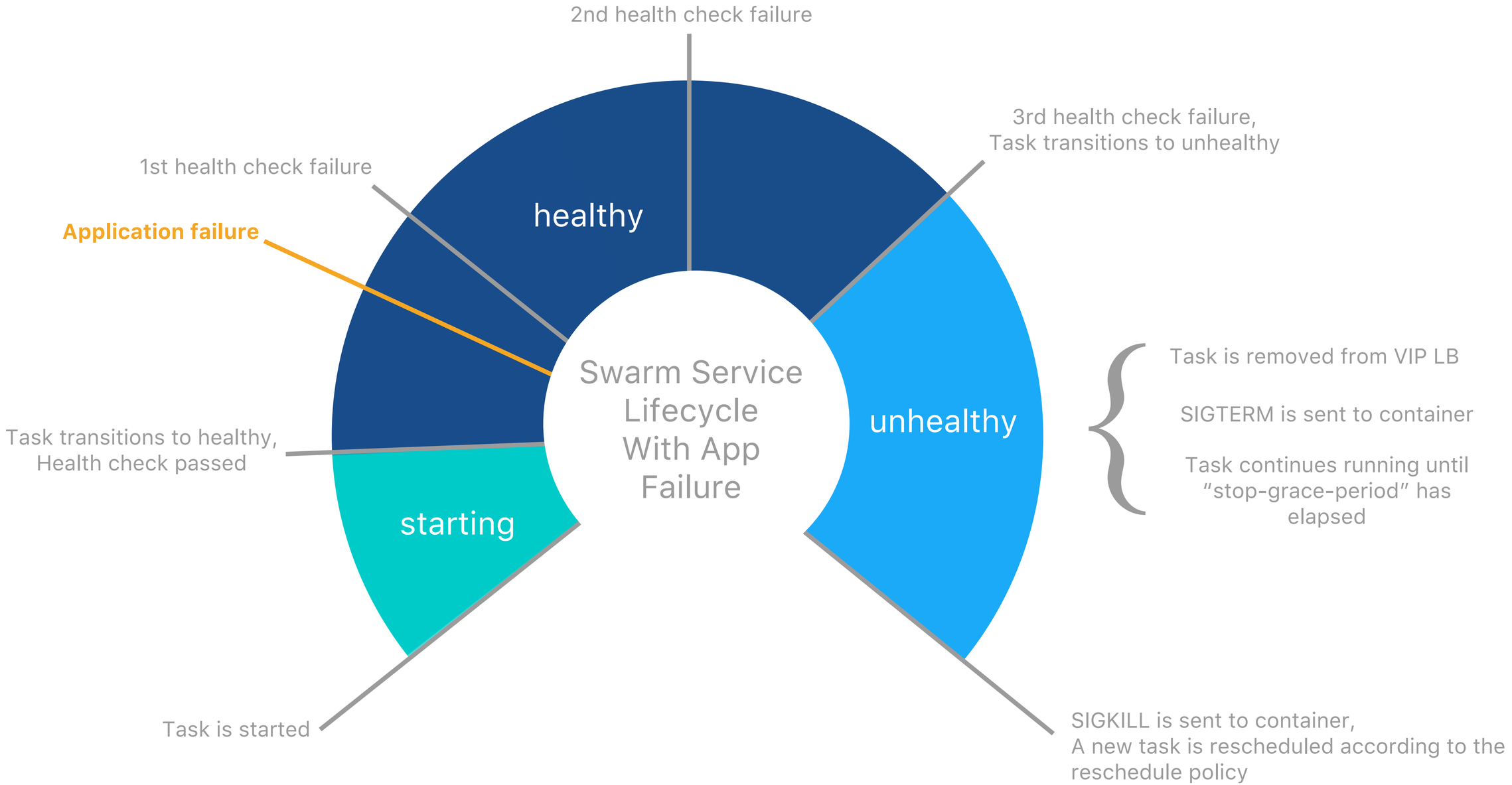
How Healthchecks Affect Updates

Batteries included, but swappable

--health-cmd	Command to run to check health
--health-interval	(default 30s) Time between running the check (ms|s|m|h)
--health-retries	(default 3) Consecutive failures needed to report unhealthy
--health-start-period	(default 0s) Time for container to start before counting to unstable
--health-timeout	(default 30s) Maximum time to allow one check to run

--stop-grace-period	Time to wait before force killing a container (ns|us|ms|s|m|h)
--no-healthcheck	Disable any container-specified HEALTHCHECK


2nd health check failure
Application recovery
Health check passed,
Failure count starts over
1st health check failure
Application failure
healthy
Swarm Service
Lifecycle
with App
Failure and
Recovery
Task transitions to healthy,
Health check passed
Health check passed
starting
Task is started


2nd health check failure
3rd health check failure,
Task transitions to unhealthy
1st health check failure
Application failure
healthy
Swarm Service
Lifecycle
With App
Failure
unhealthy
Task is removed from VIP LB
SIGTERM is sent to container
Task continues running until
“stop-grace-period” has
elapsed
Task transitions to healthy,
Health check passed
starting
Task is started
SIGKILL is sent to container,
A new task is rescheduled according to the
reschedule policy

Healthcheck Tips

- Only validate current container, nothing external or data
  - Ensure index page returns 200
  - Ensure nginx proxy returns 200 for /ping page
  - Ensure DB accepts connections, returns tmp DB
- Leave "integration healthchecks" for external monitoring
  - Ensure web API returns valid DB data
  - Ensure DB returns proper table/record count
  - Ensure web front-end can query API

Testing Updates with Healthchecks

- Run a basic service, then update it while we httping
  - > docker network create --driver overlay --attachable verse
  - > docker service create --name firefly -p 80:80 --network verse --replicas 5 --constraint "node.hostname==node1" --env DELAY_STARTUP=5000 bretfisher/browncoat:v1
- Lets watch it with httping and events
  - > docker run --rm --network 1 bretfisher/httping -i .1 -GsY firefly/healthz
  - > docker events -f service=firefly
- Lets update it and see connection failures without healthcheck
  - > docker service update --image bretfisher/browncoat:v2 firefly
- That sucked, lets update to an image with healthcheck
  - > docker service update --image bretfisher/browncoat:healthcheck firefly

Lecture Cleanup

- Stop the httping container
  - Windows: ctrl-c and docker stop <container name>
  - Linux/macOS: ctrl-c

- Remove the service and network we created

> docker service rm firefly

> docker network rm verse

Assignment: Create Healthchecks

- Use Docker Healthcheck Library
  > git clone https://github.com/docker-library/healthcheck
- Go through some examples, especially ones you know you'll use
- Try commands manually inside containers to see how they work
- Use my node sample to healthcheck using code not curl
  > git clone https://github.com/BretFisher/node-docker-good-defaults

Service Rollback

- Review Service Updates & Healthcheck lectures if needed
- 1st way it's used: manual docker service rollback <service>
  - No options, goes back to last service definition
- 2nd way: auto rollback during service update
  - The last resort if your update doesn't go as planned
  - Not the default --update-failure-action (pause)

Rollback Options During Update

Batteries included, but swappable

--rollback-delay	Delay between task rollbacks (ns|us|ms|s|m|h)
--rollback-failure-action	Action on rollback failure ("pause"|"continue")
--rollback-max-failure-ratio	Failure rate to tolerate during a rollback
--rollback-monitor	Duration after each task rollback to monitor for failure
--rollback-order	Rollback order ("start-first"|"stop-first")
--rollback-parallelism	Maximum number of tasks rolled back simultaneously

Timeline Of A Service Rollback

- Swarm will rollback N instances at a time --rollback-parallelism
- New tasks are created, and their desired state is set to **Ready**
  - Ensures resource availability (**Pending**)
  - Pulls the image if necessary (**Preparing**)
  - Creates the container ... without starting it (**Ready**)
- If a task fails to get to **Ready** state, it retries with a new task
- When tasks are **Ready**, it sets the old tasks desired state to **Shutdown**
- When the old tasks are **Shutdown**, it starts the new tasks, set to **Running**
- Then it waits for the --rollback-delay, and continues with the next task batch

Testing Rollbacks

- deploy service with healthcheck

  > docker service create --name firefly --replicas 3 bretfisher/browncoat:healthcheck

- run an update that fails

  > docker service update --image bretfisher/browncoat:v3.healthcheck firefly

- rollback manually

  > docker service rollback firefly

- run an update that fails and rolls back automatically

  > docker service update --image bretfisher/browncoat:v3.healthcheck \
  --update-failure-action rollback firefly

Lecture Cleanup

- Remove the service we created

> docker service rm firefly

Assignment: "Day 2" the Voting App

- From the course repo, use the Example Voting App stack file
  > cd example-voting-app
- The repo for those images and source code are here
  > https://github.com/BretFisher/example-voting-app
- Add features from this sections learnings
- Healthchecks and healthcheck options
- Update options
- Rollback options

Section Review: Updates and Healthchecks

- Day 1 vs. Day 2 Operations
- Rolling Service Updates
- Timeline of a Service Update
- How Healthchecks Affect Updates
- Creating Your Own Healthchecks
- Gracefully Handle Update Issues with Rollbacks