#DOCKERATH

DOCKER WORKSHOP

@AKALIPETIS

- Docker Captain and early user
- Python lover and developer
- Technology lead at SourceLair / stolos.io
- Docker training and consulting

I love automating stuff and sharing knowledge around all things containers, DevOps and optimizing developer workflows.

AGENDA

- Intro to Docker, from development to production
  - Docker, Docker Compose, Testing and Deployments
- Orchestrating containers
  - Docker Swarm, Managing a cluster, Security scanning
- Deploying complex applications
  - Multi-host networking, Storage and volumes, scaling, rolling updates

WHAT IS A CONTAINER?

CONTAINERS ARE THE USE OF A COLLECTION OF KERNEL TOOLS AND FEATURES, IN ORDER TO JAIL AND LIMIT A PROCESS ACCORDING TO OUR NEEDS AND WANTS.

WHAT IS A CONTAINER

- It’s a process
- Isolated in it’s own world, using namespaces
- Resource limited, using groups

CONTAINERS ARE NOT VMS. VMS ARE LIKE SUMMER VILLAS, CONTAINERS ARE LIKE SUPER MODERN AND EFFICIENT APARTMENTS

DOCKER CONTAINERS

- Copy on write filesystem
    - Custom OS, lightning fast
- Control over the network stack
    - Join multi-host SDNs
- Volume management and mounting
    - Take your data with your

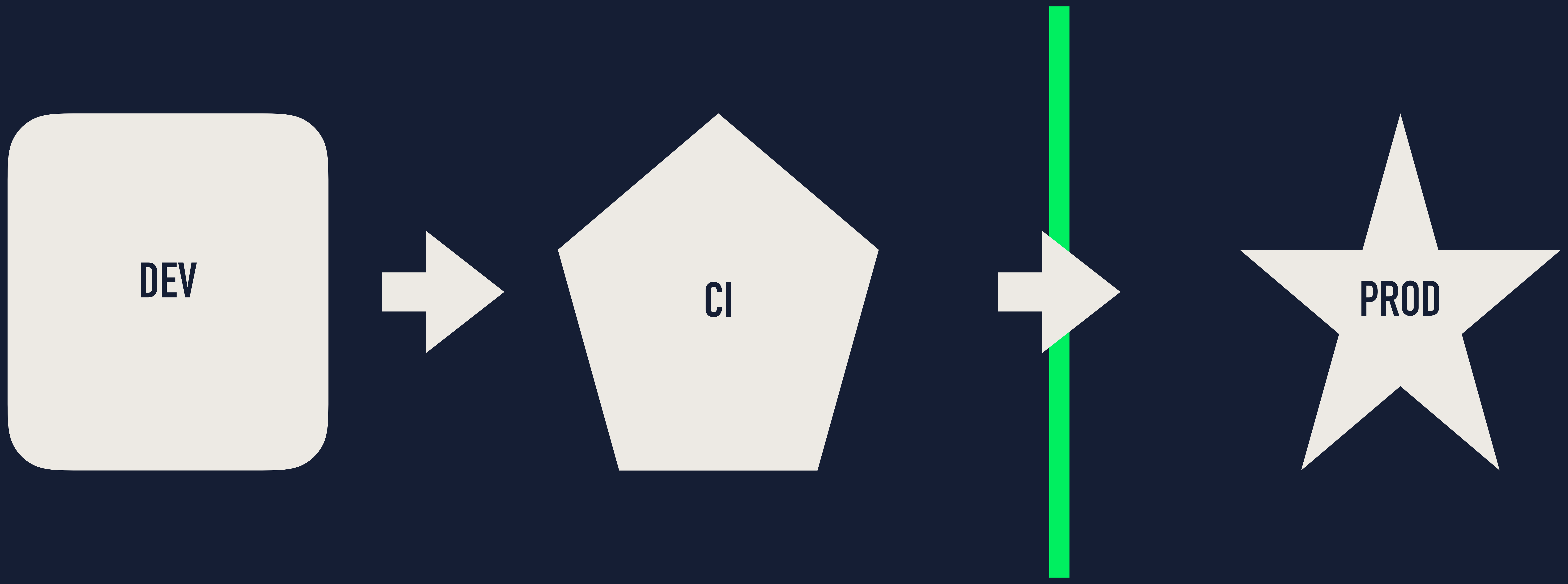
THE COMPONENTS

- runc - the runtime
  - Makes sure your applications run the
- containerd - the container manager
  - Manages all containers in a node
- Docker - the orchestrator
  - Orchestrates containers in multiple nodes

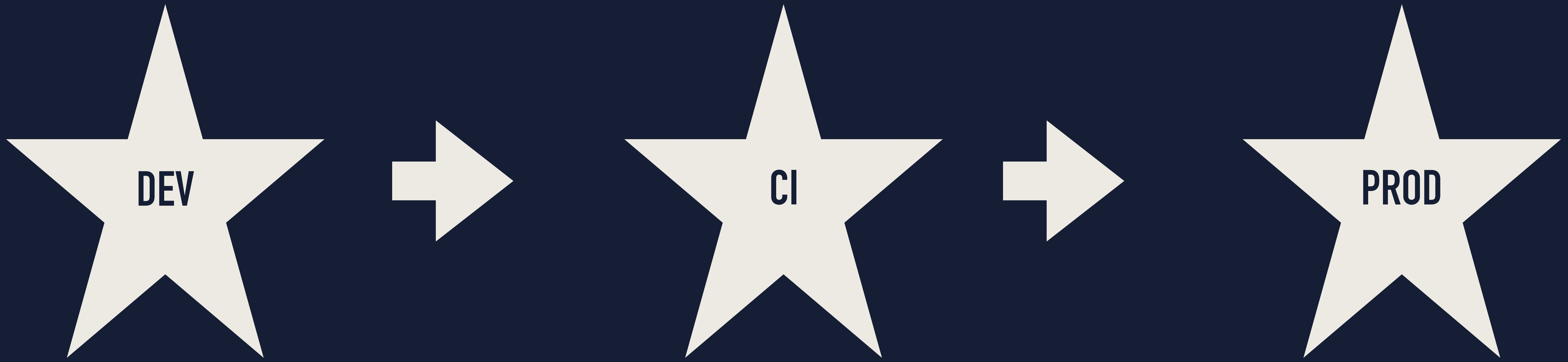
DOCKER COMPOSE

BEST PRACTICES FROM DEV TO PROD

THE PROBLEM



THE DREAM



OPTIMIZING YOUR DELIVERY PIPELINE

- Same runtime in development, CI and production
- Use the same declarative format all the way
  - Docker Compose works both in one node and Docker Swarm
- Focus on what you do best
  - Developers should code
  - Ops should manage infrastructure
  - Application management is left to the Swarm

DEVELOPING ON YOUR LOCAL MACHINE

- Use Docker Compose
  - Try to have a similar set of files for both development and production
- Get up and running quickly with Docker for Mac / Windows
  - This gives you the whole toolchain, Docker and Docker Compose
- Have a series of Dockerfile
  - For example Dockerfile and Dockerfile.dev that derives from Dockerfile

DEV TOOLING

- Docker for Mac/Windows
  - Native support for the OS
- Docker Compose
- Auto reloading web servers
  - Nodemon, gin, Django, Flask, etc
- docker-compose up

BENEFITS

- All your dependencies are version controlled
  - Database/cache versions
- You still use your favourite local editor for development
  - Volume mounting inside the container

SETTING UP YOUR CI

- There are already CI systems that use a Compose-like syntax
  - Drone and Codeship to name a few
- You can still use your favourite CI
  - Just do a docker-compose run <service> make test

BENEFITS

- Use the same, production ready stack for your tests
  - Having a MongoDB replica set per test is super cheap
- Test against different software versions
  - Test against different database or language versions painlessly
- Spawn multiple, parallel and disposable testing stacks
  - After your tests run, your database and data can be deleted and rebuilt

DEPLOY DOCKER STACKS WITH COMPOSE

- You can deploy a docker-compose.yml file as a Docker stack
  - docker stack deploy -c docker-compose.yml mystack
- New v3 Docker Compose format
  - Removes any non-portable parameters, ie volumes_from
  - Adds new deployment section, allowing for replicas, replication mode, rolling updates, etc
- Improve dev to production pipeline with same tooling

DEPLOYING WITH DOCKER COMPOSE

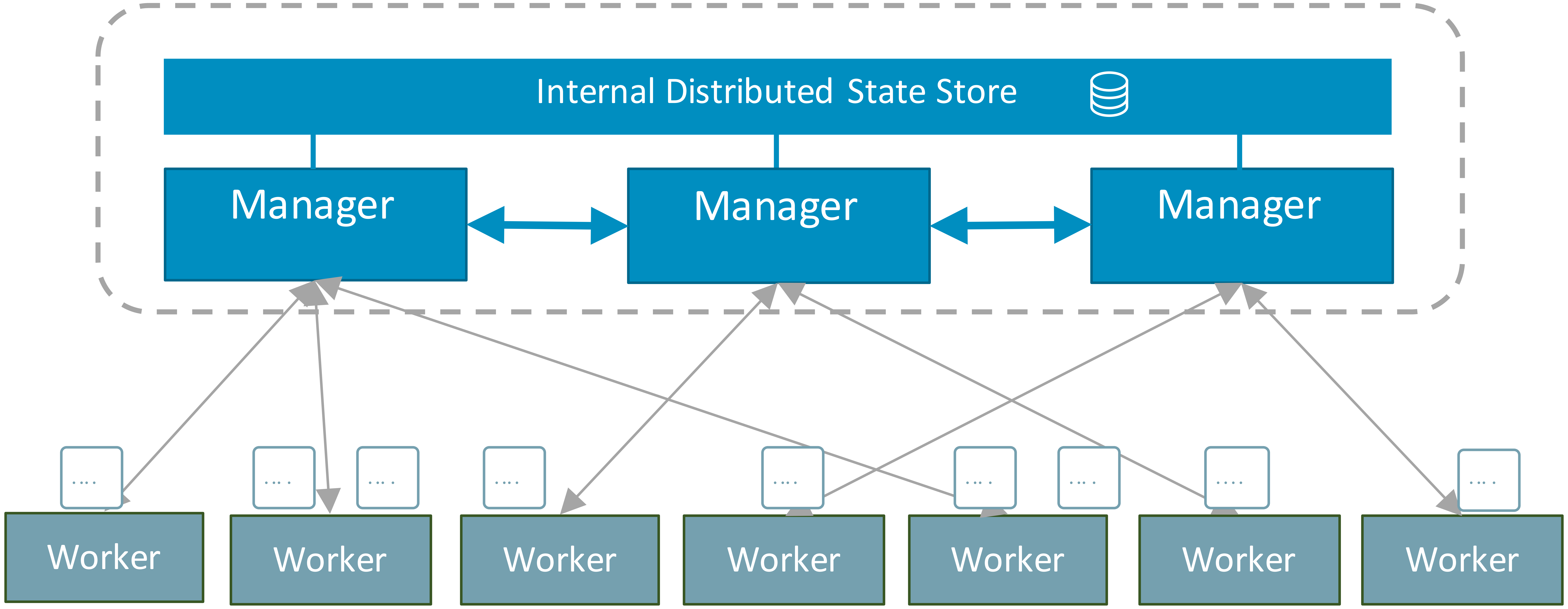
- Version control your complete system
  - Networks, volumes, databases, etc
- Use the same stack that you used for dev and CI
  - Forget “this worked on my machine”
  - If tests passed, then your code is ready

LET'S TAKE A BREAK

DOCKER SWARM

DOCKER IS AN OPERATING SYSTEM FOR YOUR DATA CENTER

DOCKER SWARM TOPOLOGY



LET'S CREATE OUR FIRST SWARM

LET'S CREATE OUR FIRST SWARM

- http://play-with-docker.com
- Create 3 nodes
- docker swarm init
- docker swarm join in nodes 2 and 3

KEY CONCEPTS

- Node
  - The servers in your cluster, either managers or workers
- Services
  - A group of tasks that share the same configuration
- Tasks
  - Mapped 1-1 with containers, but could be any deployable unit

DOCKER SWARM SERVICES

- Declarative definition
  - Replicas
  - Global vs replicated
  - Rolling updates

DOCKER SWARM SERVICES

- docker service create --name=nginx --publish=8080:80 nginx
- docker service scale nginx=6
- docker service create –name=nginx-g --mode=global --publish=8081:80 nginx

INGRESS LOAD BALANCING

- Each service in the Swarm gets a virtual IP
  - The Swarm makes sure connections to this internal IP are routed to the correct container, in any host in the Swarm
- Multi-host networking is made with pluggable network drivers
- If desired, a port is opened to each node of the Swarm
  - Connections to this port are routed to the service virtual IP, at a defined port

MULTI-HOST NETWORKING / THE OVERLAY DRIVER

- The Overlay driver uses packet encapsulation
- Requires that all nodes in the cluster are reachable by all nodes
- Optionally, supports encryption

- docker network create --opt encrypted --driver overlay my-multi-host-network
- docker network create --attachable --driver=overlay attachable-net

ENCRYPTION AT REST

- Your swarm is encrypted at rest
  - docker swarm init --autolock
  - docker swarm unlock
  - docker swarm update --autolock=true
  - docker swarm unlock-key --rotate

SECRETS

SECRETS AS A FIRST CLASS CITIZEN

- Manage sensitive data within containers
  - Database passwords, SSH keys, TLS certificates
- Mounted as an in-memory filesystem to the container
  - cat /run/secrets/my_secret_data
- Encrypted at rest, as they're part of the Swarm Raft log

WHY USE SECRETS

- Encrypted at rest and while in motion
    - Use Raft and
- Available only to the worker running a service's task
    - Secrets are sent as part of the payload for a service task
- Minimize the possible attack surface
    - If a worker doesn't run a task of a service, the secret is never made available

SECRETS AS A FIRST CLASS CITIZEN

- docker secret create my-secret -
- docker service create --secret=my-secret nginx

PLUGINS

SAY HELLO TO DOCKER PLUGINS V2

- Downloadable from Docker Hub
- Give only the required permissions to plugins
  - They run inside runs containers
- They're managed by the engine (soon by the Swarm)
  - docker plugin install/uninstall
  - docker plugin enable/disable

OUT OF THE BOX OFFERINGS

- Docker for AWS / Azure
- Docker Datacenter
- Docker Cloud

VERIFYING PAYLOAD RUN IN A SWARM

- Images is swarm are identified by SHA256 hashes
  - If you have a verified hash, the image is verified
- Swarm can use Docker content trust to verify an image tag
  - Create a trust chain, leading to a trusted key

SCANNING IMAGES FOR SECURITY VULNERABILITIES

- Images can be also scanned for security vulnerabilities
  - Docker EE provides
  - Images in the Docker store are scanned
- There are open source projects that you can use internally
  - coreos/clair

IS THIS SECURE?

YES IT IS, IF YOU KNOW WHAT YOU'RE DOING

- Docker has the tooling to reduce your attack surface
  - allow write only to the directories you want
  - clean API for whitelisting tools like AppArmor
- Process isolation
  - chances to escape the container are small, especially for non-root users
  - namespaces help you "hide" host information, like mount points or network

DECLARATIVE, SELF-HEALING INFRASTRUCTURE

INFRAKIT

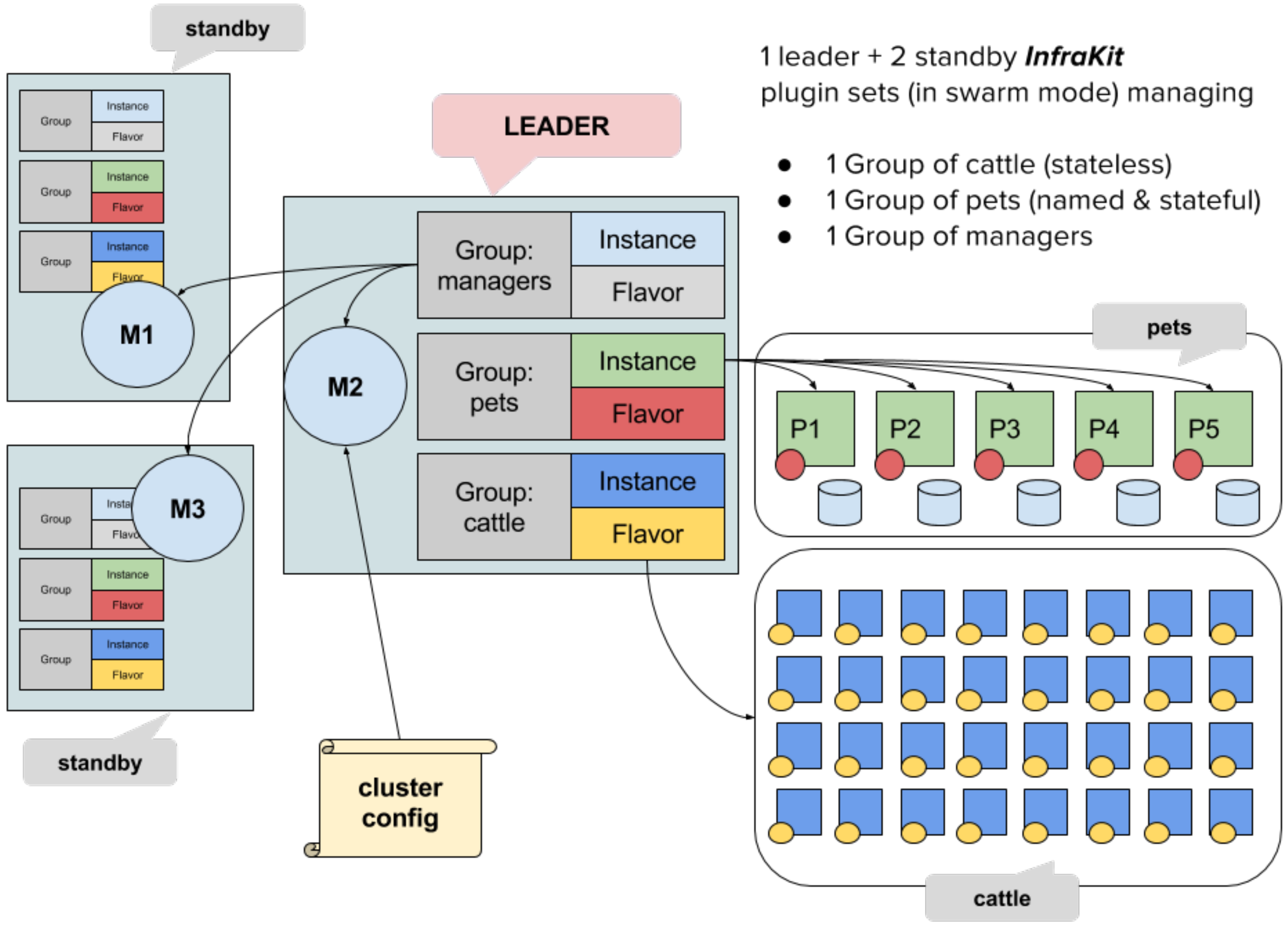
WHAT IS INFRAKIT

- A toolkit for building declarative, self-healing infrastructure
- JSON configuration for desired infrastructure state
  - VM image, instance type, etc
  - Groups properties like size, identifiers, cattle VS pet
- Active monitoring
  - Check and ensure cluster state

WHAT'S INCLUDED

- Basic principles, like create, scale, destroy and rolling update
- Abstractions for base entities
  - Group
  - Instance
  - Flavor
- Plugins
  - Now go binaries, later on runc containers

ARCHITECTURE



1 leader + 2 standby ***InfraKit*** plugin sets (in swarm mode) managing

- 1 Group of cattle (stateless)
- 1 Group of pets (named & stateful)
- 1 Group of managers

LET’S TAKE A BREAK

COMPARING ORCHESTRATORS

STORAGE PLUGINS

STORAGE PLUGINS

- Manage your data for you
    - The data travels with the container
- Allows for databases and other state full services to be deployed in a cluster
- Popular backends include
    - Block storage, that gets reattached
    - Network filesystems that provide volumes to the correct node

STORAGE PLUGINS

- Common storage plugins
  - RexRay
    - Block storage, NFS, others like S3FS
  - Cloudstor
    - Azure and AWS NFS-like FS, block storage coming soon
  - Convoy
    - Device mapper backed, with easy backups

LET'S DEPLOY A MONGO REPLICA SET

LET’S DEPLOY AN AGENT SERVICE

LET'S DEPLOY AN AGENT SERVICE

- deploy:
-     mode: global

ALLOCATING RESOURCES

ALLOCATING RESOURCES

- There are two limits we need to control
    - The maximum limit that a container will use
    - The guaranteed allocation to exist for a container to be scheduled

- resources:
-         limits:
-           memory: 4G
-         reservations:
-           memory: 2G

HEALTHCHECKS

HEALTH CHECKS

- Route requests to healthy containers only
- Rolling updates - wait for container to become healthy
- healthcheck:
-     test: ["CMD", "curl", "-f", "http://localhost:5000"]
-     interval: 10s
-     timeout: 10s
-     retries: 3

ROLLING UPDATES

ROLLING UPDATES

- Control how your application will be updated
- Don’t let your users down while updating
- Easy to revert
- update_config:
-         parallelism: 2
-         delay: 10s

MANAGING NETWORKS

MANAGING NETWORKS

- Partition networks, according to use case
  - Frontend
  - Backend
  - Mongonet
- Join only the minimum amount of networks needed
- Don’t expose services to others that don’t need them

#DOCKERATH

THANKS