@STEVESLOKA

SHIP IT! CONTAINERIZED CLOUD-NATIVE DEPLOYMENTS

OVERVIEW

- Reality Check
- What is “Cloud Native”?
- Deploying locally with Docker && Compose
- Deploying with Kubernetes
- Developing with Emmie
- Summary

ABOUT ME

SOFTWARE ARCHITECT / ENGINEER

Without software: Phones don't ring. Cars don't start. Planes don't fly. Bombs don't explode. Ships don't sail. Ovens don't bake. Garage doors don't open. Money doesn't change hands. Electricity doesn't get generated. And we can't find our way to the store. Nothing happens without software.

Uncle Bob

"CLOUD NATIVE"

DEFINE “CLOUD”

noun

1. a visible collection of particles of water or ice suspended in the air, usually at an elevation above the earth's surface.

dictionary.com

DEFINE “NATIVE”

adjective

1. being the place or environment in which a person was born or a thing came into being

dictionary.com

Noun

1. The place or environment in which a person was born, visualized as collection of particles of water or ice suspended in the air.

Steve Sloka

MADE UP FACTS

SOUNDED SOPHISTICATED


BUZZWORDS
BUZZWORDS EVERYWHERE
quickmeme.co


hello.

LET'S TRY TO DEFINE

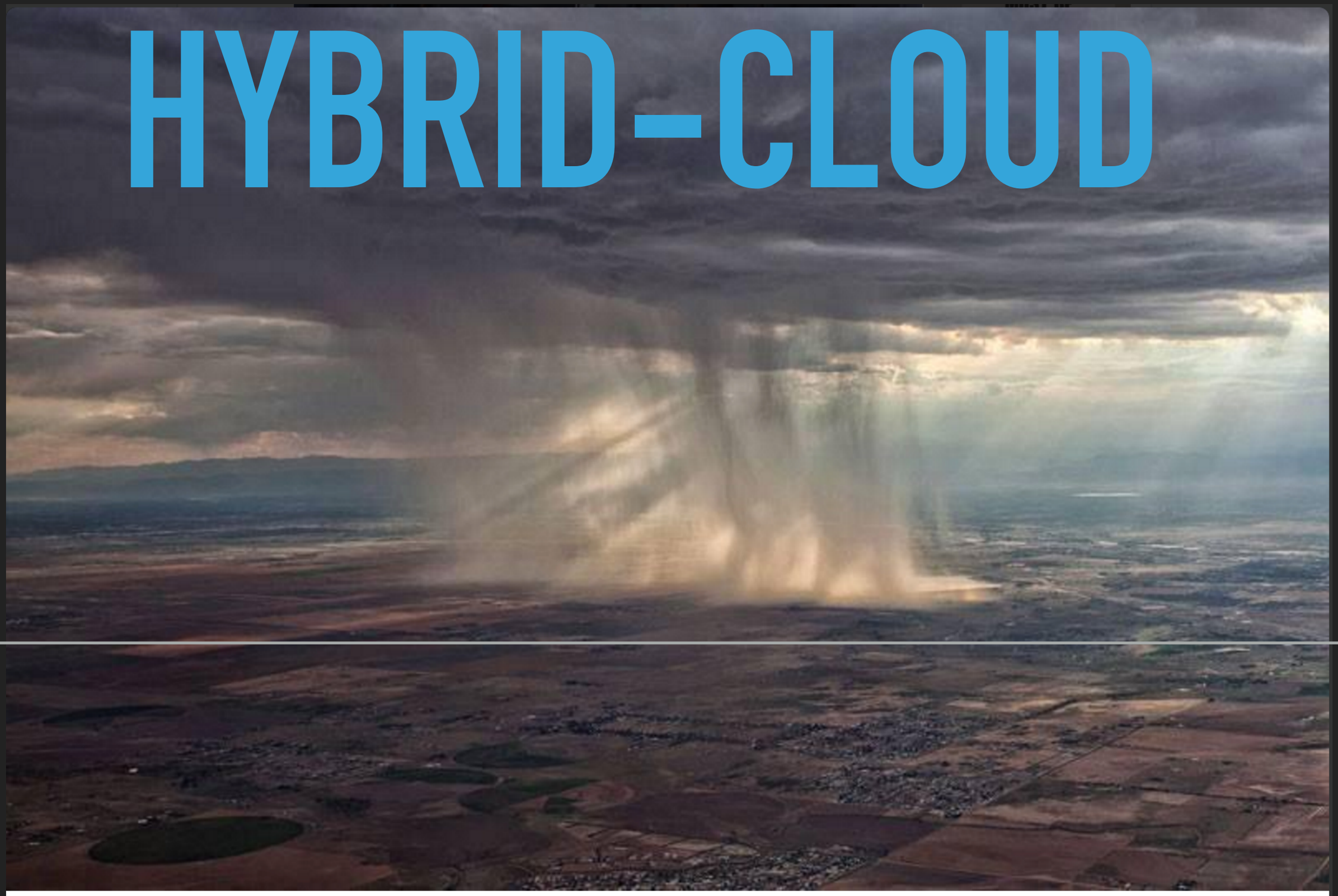
CLOUD NATIVE

PUBLIC CLOUD

PRIVATE CLOUD

RANCHER

HYBRID-CLOUD



Werner Vogels @Werner · 2h

Just found a close to perfect image to illustrate hybrid clouds :-)

55 65

THE OPERATIONS STACK

APPLICATION OPS

CLUSTER OPS

KERNEL/OS OPS

HARDWARE OPS

~ Brendan Burns

> How would you design your infrastructure if you couldn't login. Ever.
>
> Kelsey Hightower

12 FACTOR

12-FACTOR OVERVIEW

- **Codebase**
  One codebase tracked in revision control, many deploys
- **Dependencies**
  Explicitly declare and isolate dependencies
- **Config**
  Store config in the environment
- **Backing Services**
  Treat backing services as attached resources
- **Build, release, run**
  Strictly separate build and run stages
- **Processes**
  Execute the app as one or more stateless processes
- **Port binding**
  Export services via port binding
- **Concurrency**
  Scale out via the process model
- **Disposability**
  Maximize robustness with fast startup and graceful shutdown
- **Dev/prod parity**
  Keep development, staging, and production as similar as possible
- **Logs**
  Treat logs as event streams
- **Admin processes**
  Run admin/management tasks as one-off processes

DEV/PROD PARITY

- The time gap: A developer may work on code that takes days, weeks, or even months to go into production.
- The personnel gap: Developers write code, ops engineers deploy it.
- The tools gap: Developers may be using a stack like Nginx, SQLite, and OS X, while the production deploy uses Apache, MySQL, and Linux.

TYPICAL ENVIRONMENTS

DEV

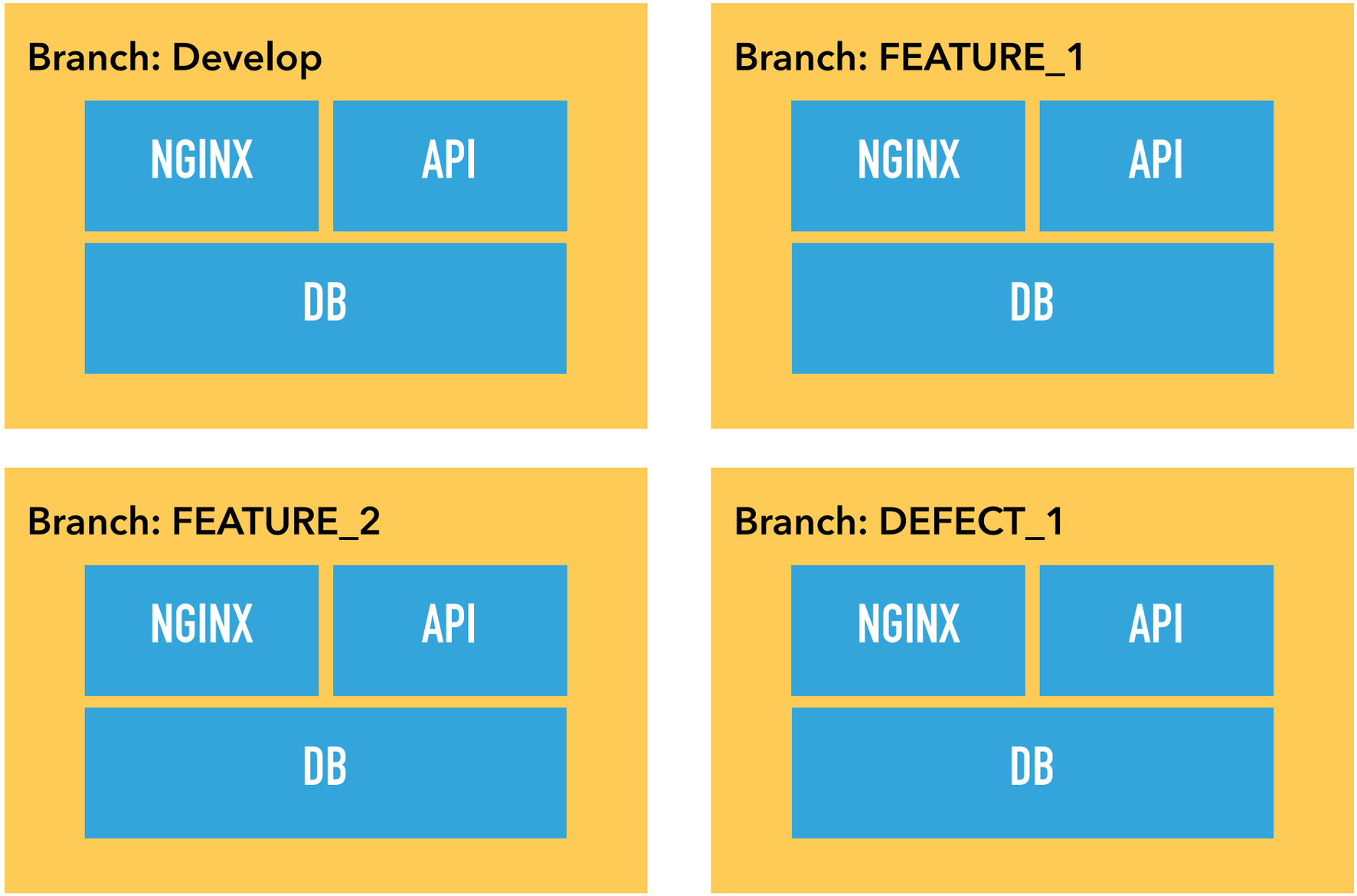
TEST

STAGE

QA

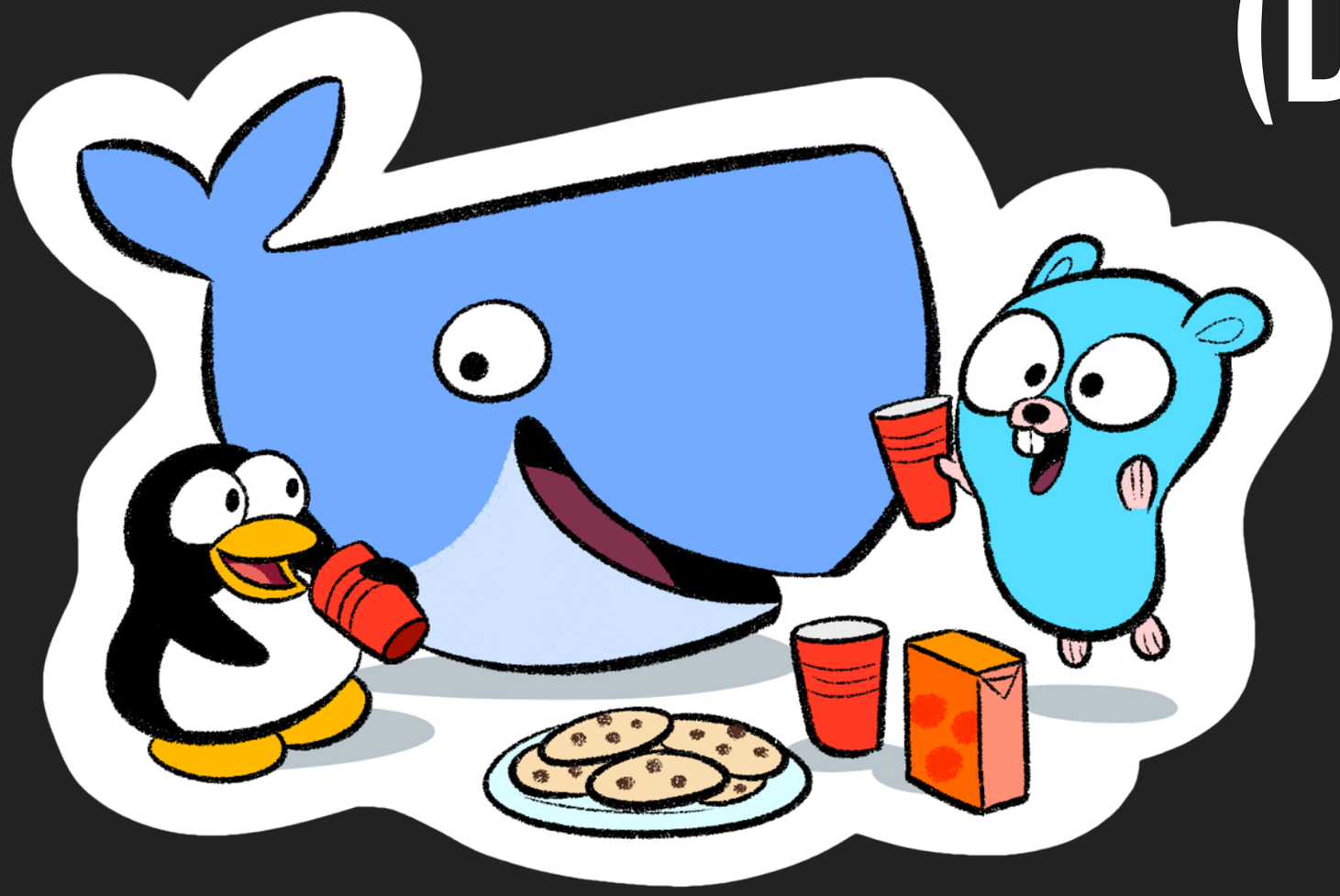
PROD

WHILE(TRUE) { YOU.GET-ENVIRONMENT(); }




YOU GET AN ENVIRONMENT
2004
generator-meme.com
AND YOU GET AN ENVIRONMENT

CONTAINERS (DOCKER)



PHP CEO
@PHP_CEO

Follow

OK, SO I'VE INSTALLED DOCKER

ARE WE DEVOPS YET OR DO I NEED TO INSTALL CONTAINERS TOO

ALSO WHERE DO I DOUBLE CLICK TO MAKE IT CLOUD AWARE

RETWEETS 1,086 LIKES 912

5:04 PM - 1 Dec 2015

WHY CONTAINERS?

- Abstraction
- Dependencies
- Speed
- Consistency
- Portability

DOCKER TIPS

- Same “app” container is used in all environments
- NEVER hard code config / passwords / into containers*
  - Config Containers*
    - Use centralized config servers
    - Use Environment Variables
- Containers do one thing
- Docker tag “latest” is not a version!

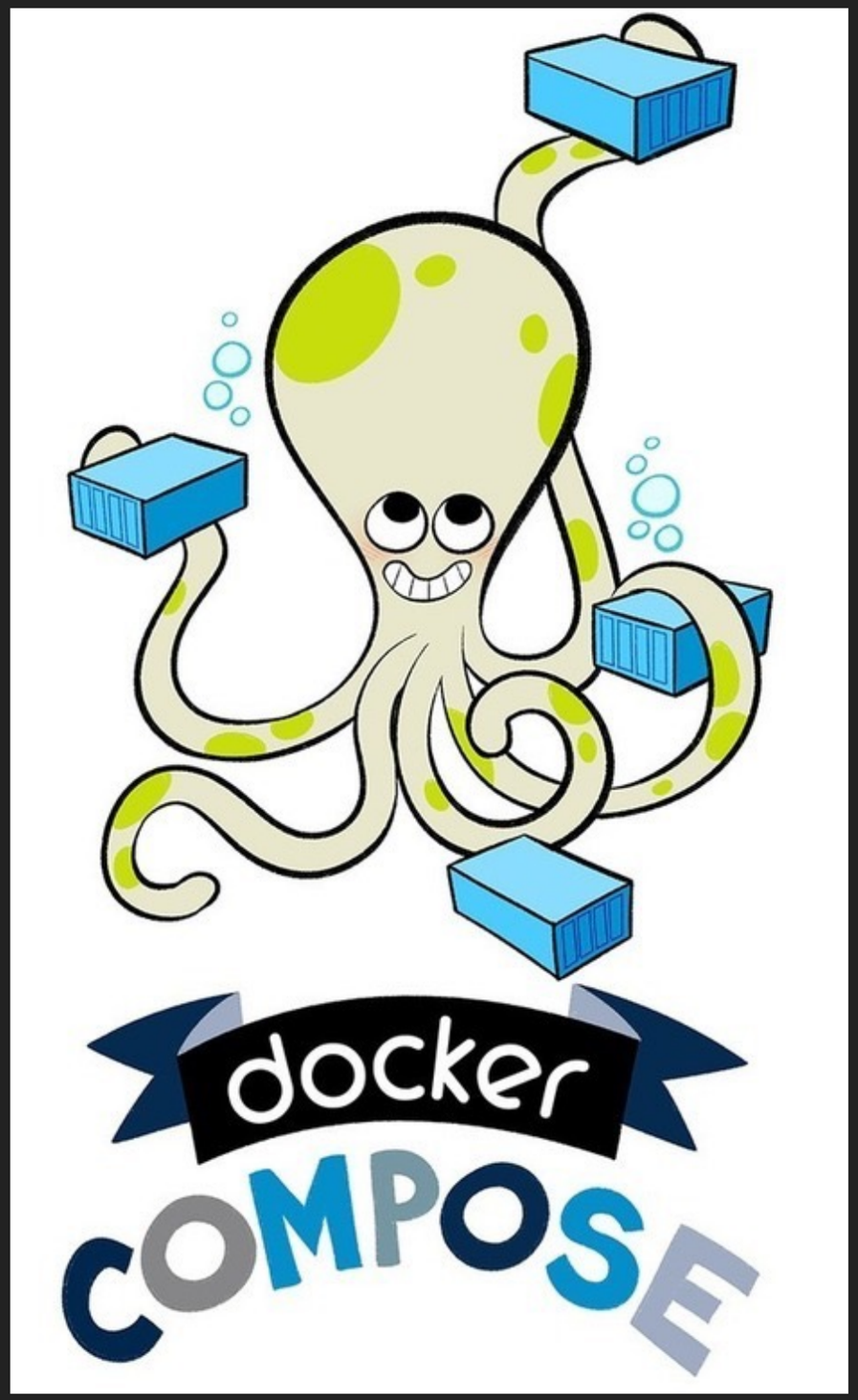
DOCKER CLI

```
docker run --restart=always --name ${CONTAINER_NAME}_api -d \
  --link ${CONTAINER_NAME}_db:apidb \
  --link ${CONTAINER_NAME}_oauth:oauthapi \
  --link ${CONTAINER_NAME}_config_server:configserver \
  -e "spring.cloud.config.uri=http://configserver:8888" \
  -e "spring.cloud.env=development" \
  -e "spring.cloud.config.label=${CONFIGURATION_BRANCH_NAME}" \
  -e "upay-return-url=http://server:${DOCKERPORT}/api/payment/summary" \
  -e "upay-cancel-url=http://server:${DOCKERPORT}/api/payment/cancel" \
  docker-repo/anywherecare/rest-api:${CONTAINER_NAME}
```

DEPLOY ISOLATED ENVIRONMENTS WITH DOCKER

- Create isolated environments by uniquely naming containers (-–name)
- Don’t define host port and Docker will auto generate random port (-p 80)

TYPICALLY END UP SCRIPTING SINCE IT GETS COMPLEX QUICKLY

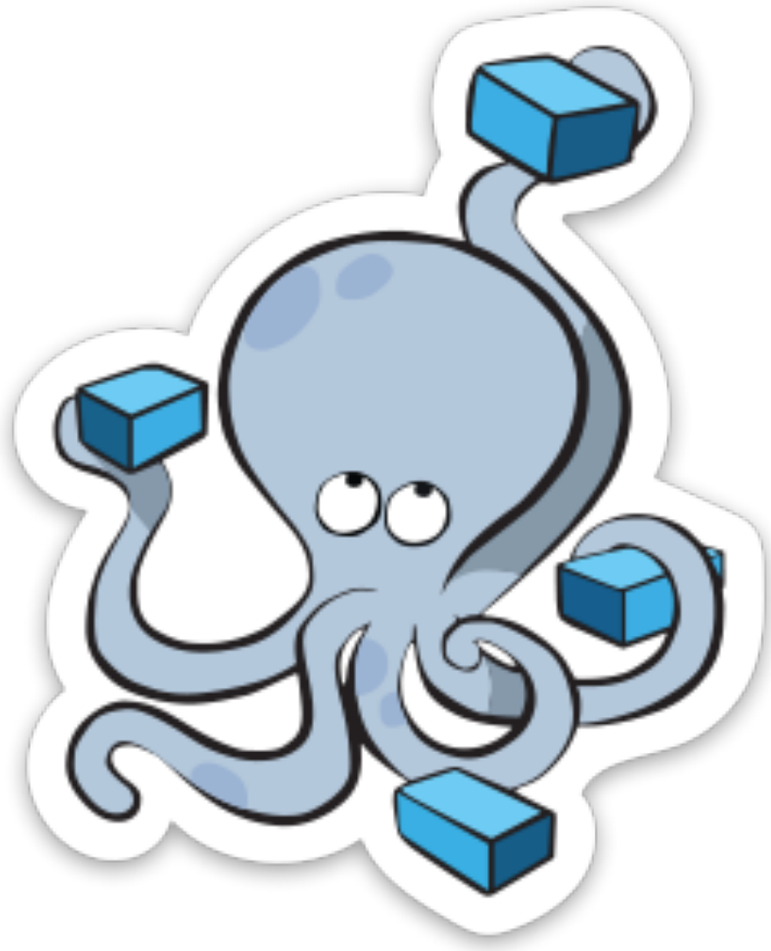


TO THE RESCUE!

DOCKER-COMPOSE

DOCKER-COMPOSE

```
nginx:
  build: .
  volumes:
    - app:/src/app
  ports:
    - 80
  links:
    - db
db:
  image: mysql
```



DOCKER-COMPOSE TIPS

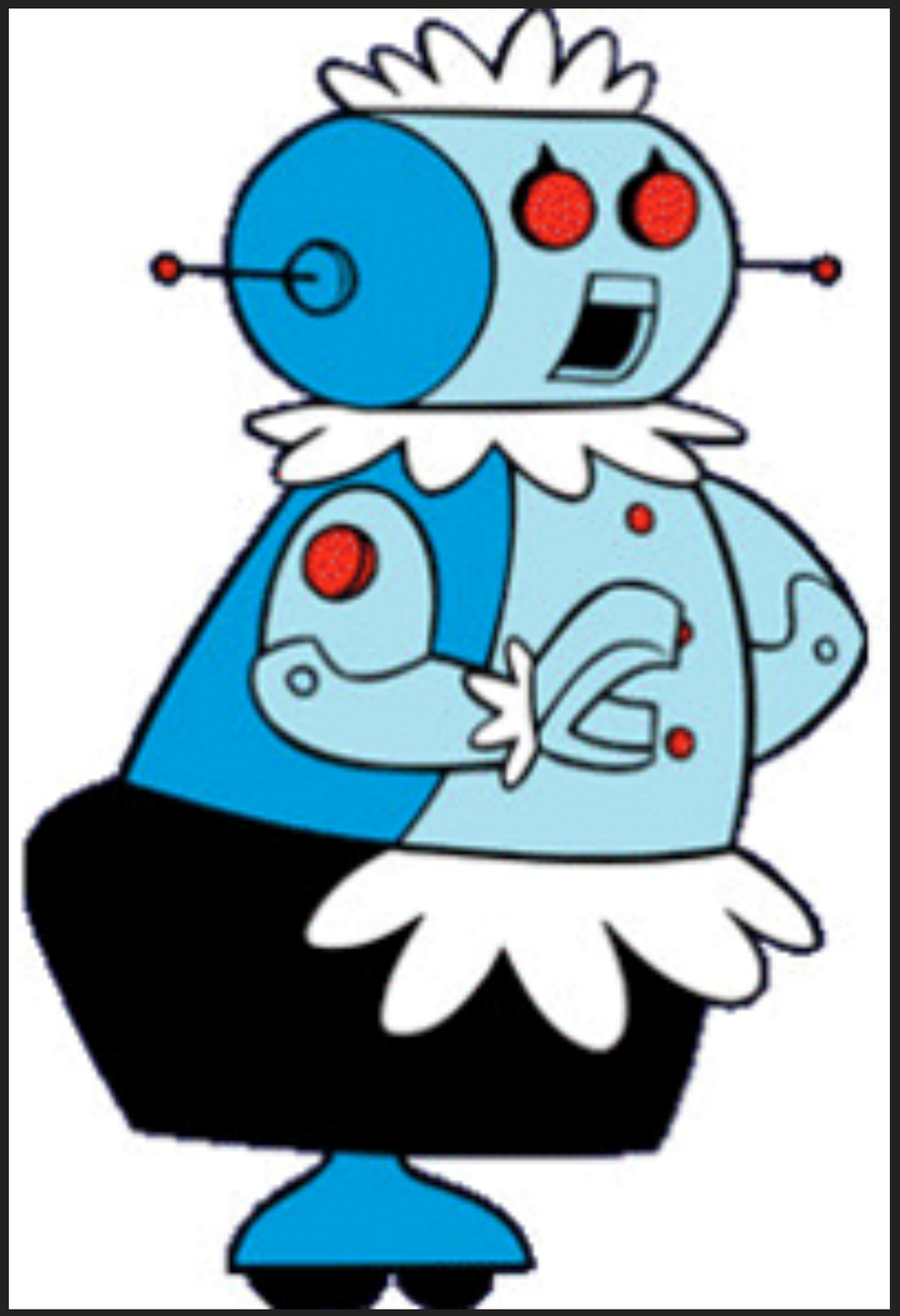
- {{Same rules as Docker}}
- Change “project name” for each deployment so to make unique environments (-p FEATURE_NAME)
  - docker-compose up -p MY_SWEET_FEATURE

"CLOUD-NATIVE"

- Containerize Everything*
  - Config files should be optional
  - Use env vars for config
- Eliminate the need to deploy services in a specific order (Exponential Backoff)
- Decouple all the things!
- Rolling back should be as easy as pushing new

"CLOUD-NATIVE" (CONT)

- Keep environments consistent
- Centralize logging
- Automate builds of binaries / docker images
- Automate everything! (All the stacks)



API'S

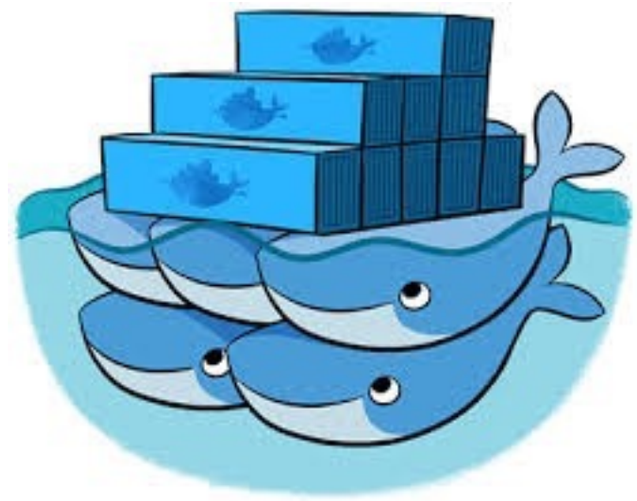
ARE AMAZING!

1 DOCKER HOST =

2+ DOCKER HOSTS = HARD

CONTAINER MANAGEMENT OPTIONS:

Nomad



MESOSPHERE

#!/bin/bash

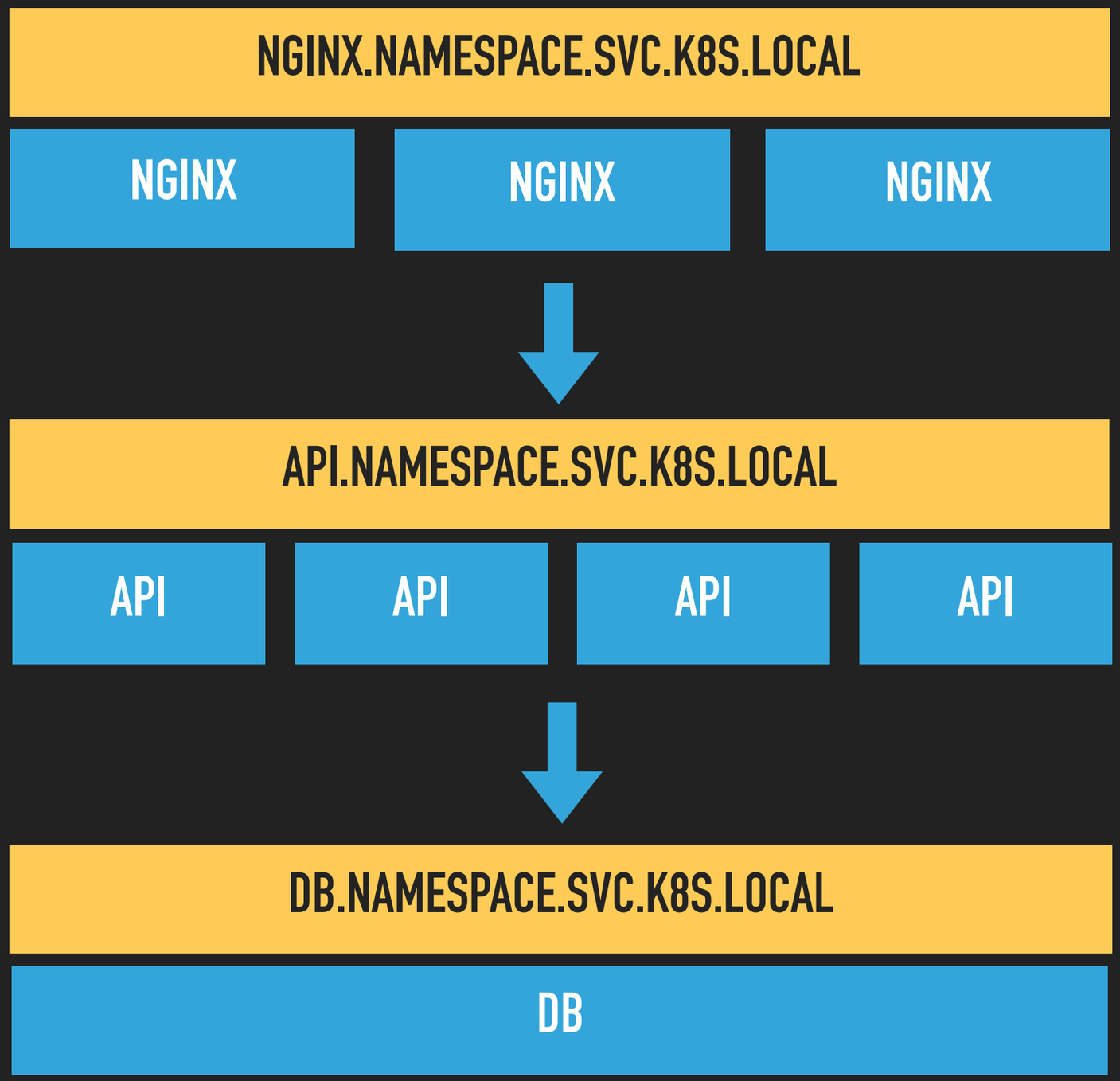
CoreOS

KUBERNETES

- Schedules docker containers to nodes in a cluster of servers
  - GIFEE*
- Replication Controllers
  - Pod
- Services
  - Service Discovery
- Namespaces

*GIFEE (GOOGLE-LIKE INFRASTRUCTURE FOR EVERYONE ELSE)

KUBERNETES SAMPLE



NAMESPACES

- A mechanism to partition resources created by users into a logically named group
- Allows for work to be done in isolation
- Each namespace is given its own:
    1. resources (pods, services, replication controllers, etc.)
    2. policies (who can or cannot perform actions in their namespace)
    3. constraints (this namespace is allowed this much quota, etc.)

EMMIE

AUTOMATE DEPLOYMENTS BASED ON GIT FEATURE BRANCHES

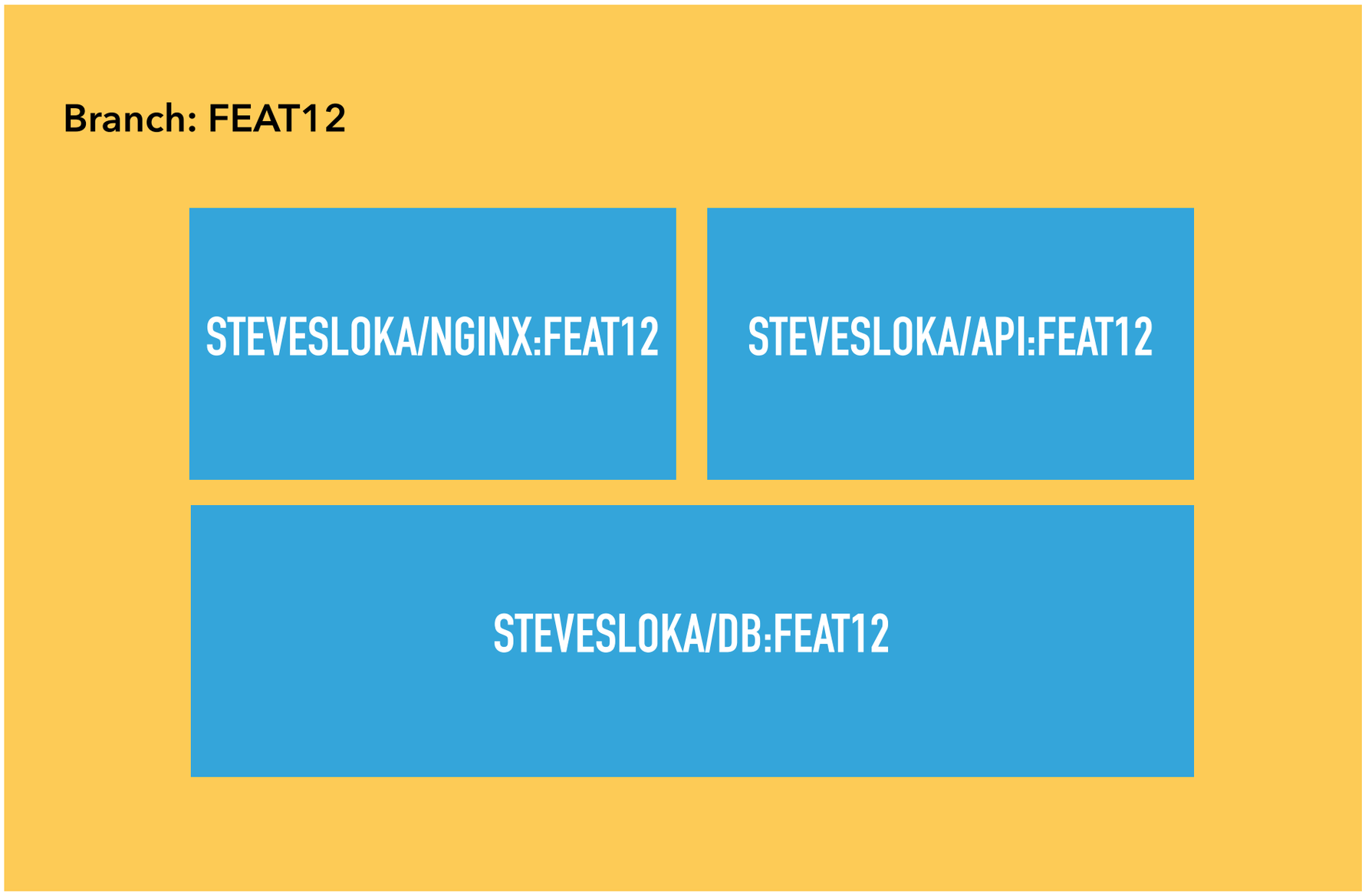
WHY ANOTHER TOOL?

- Allows to QA feature branches with the same prod stack
- Allows for a cleaner “develop” branch
- Try out features and validate in a real environment before merging
- Stop the requirement for “static” environments
  - Speed up deployments
  - No need to keep environments in sync

EMMIE REQUIREMENTS

- Working Kubernetes cluster 
- Docker images built and taggged with branchName
  - stevesloka/web:US1234_addlogging
- Deploy **entire app** to a template namespace. This can be a new namespace or configured to be "develop" branch

EXAMPLE


Branch: FEAT12
STEVESLOKA/NGINX:FEAT12
STEVESLOKA/API:FEAT12
STEVESLOKA/DB:FEAT12

EMMIE ARCHITECTURAL OVERVIEW:

Source Control

Continuous Integration

Docker Registry

You

Kubernetes


DEMO TIME
GUESS YOU CAN SAY THINGS
ARE GETTING PRETTY SERIOUS

THANK YOU!

- @stevesloka
- steve@stevesloka.com
- github.com/upmc-enterprises/emmie
- enterprises.upmc.com

- Kubernetify All The Things
  Friday 12:15pm - Rosewood