Building a Kubernetes App with Amazon EKS

Laura Frank Tacho
Director of Engineering, CloudBees
@rhein_wein

We'll Cover:

- What Amazon EKS is, and how it differs from other Kubernetes offerings
- Requirements for running an EKS cluster
- Automating app deployment to EKS with CodeShip, a CI/CD tool
- CI/CD best practices

EKS is a managed Kubernetes offering from AWS

CloudBees CodeShip is a customizable CI/CD engine designed with containerized applications in mind

Local Environments

Minikube
Docker for Mac
Docker for Windows
play-with-k8s.com

Managed Kubernetes platforms offered by cloud providers

GKE, AKS, EKS

Cloud agnostic* managed Kubernetes

Rancher Kubernetes Engine
Docker Enterprise Edition
Joyent Triton
RedHat OpenShift
CoreOS Tectonic (now part of RedHat)

Kubernetes provides a shared standard for declaring application configuration, making your containerized apps portable.

CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape l.cncf.io has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer

cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider

cncf.io/kcsp

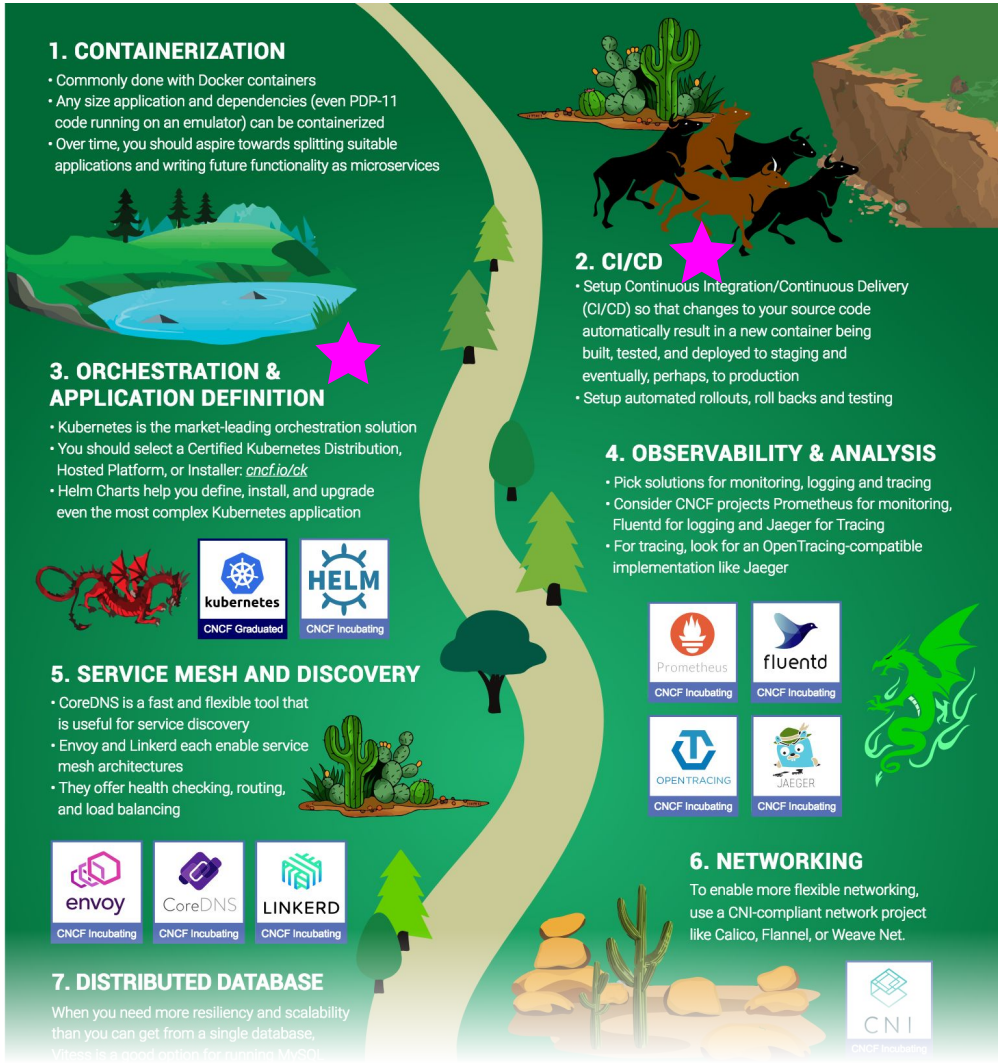
C. Join CNCF's End User Community

For companies that don't offer cloud native services externally

cncf.io/enduser

WHAT IS CLOUD NATIVE?

Cloud-native technologies, such as containers and microservices, empower organizations to develop and deploy scalable, agile applications and services in dynamic, distributed environments. By taking into account these characteristics, such systems are designed to be resilient, elastic, and loosely coupled, via manageable abstractions and declarative APIs, thereby enabling effective,



landscape.cncf.io

Managed, not magic

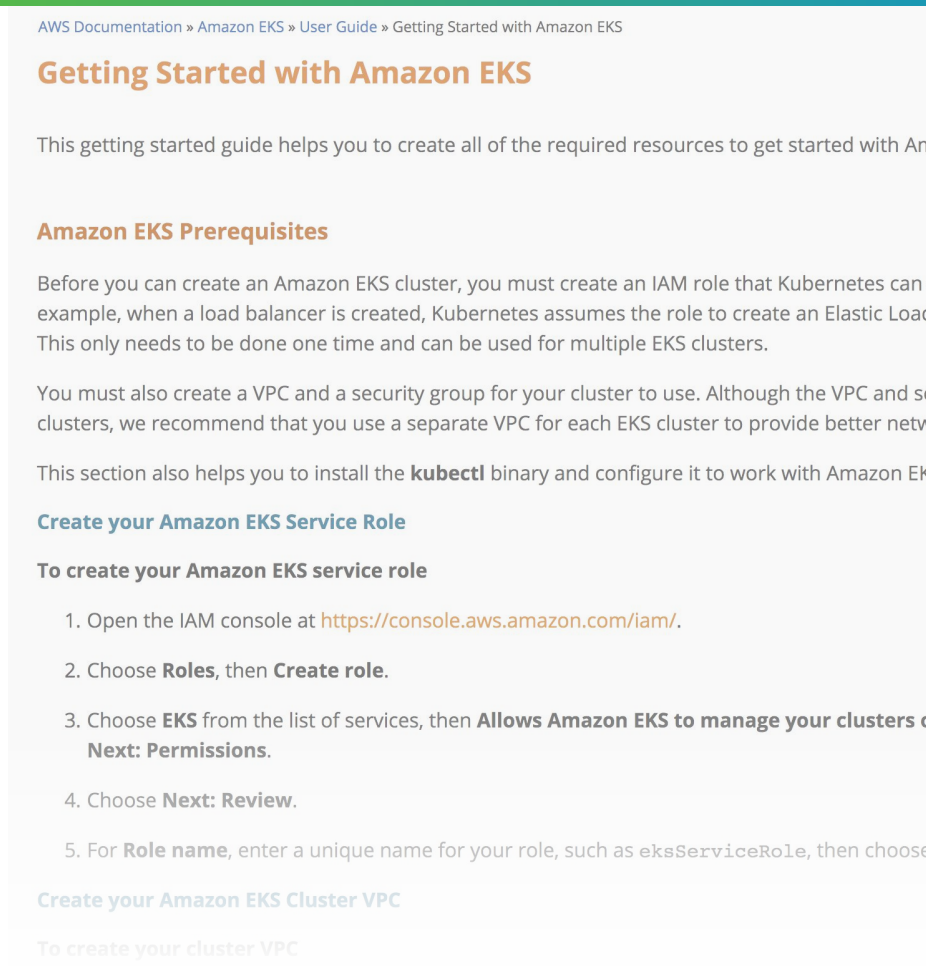
AWS docs are great, but not everything is done for you

Prerequisites:

- Basic understanding of IAM
- Able to use provided templates with CloudFormation
- Understanding of EC2 resource types

AWS CLI skills not necessary, but **helpful**

Basic understanding of kubectl is **necessary**



An Even Quicker Quickstart Guide

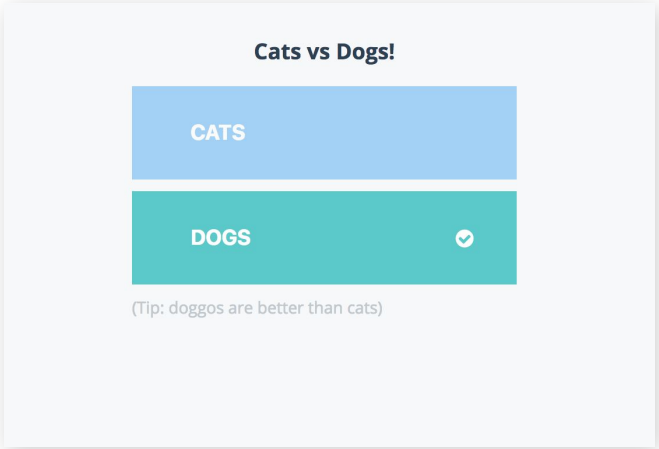
1. Create your Amazon EKS service role in the IAM console
2. Create a VPC to use with your cluster. You can use a provided CloudFormation template for this. Note that EKS is only available in `us-west-2` and `us-east-1`.
3. Install `kubectl` and `aws-iam-authenticator` for local access
4. Create your EKS cluster either via the GUI or the CLI
5. Configure access to your cluster locally
6. Launch worker nodes via CloudFormation

EKS + Terraform

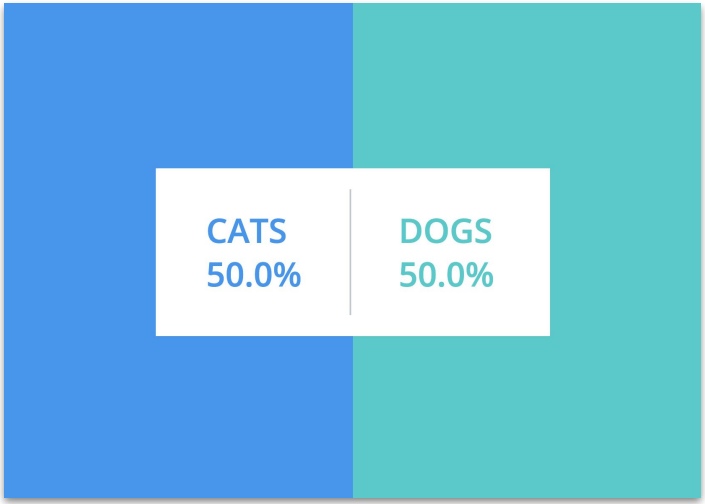
You can stand up your cluster using Terraform

Guide is available at
https://www.terraform.io/docs/providers/aws/guides/eks-getting-started.html

Sample App: Cats vs Dogs



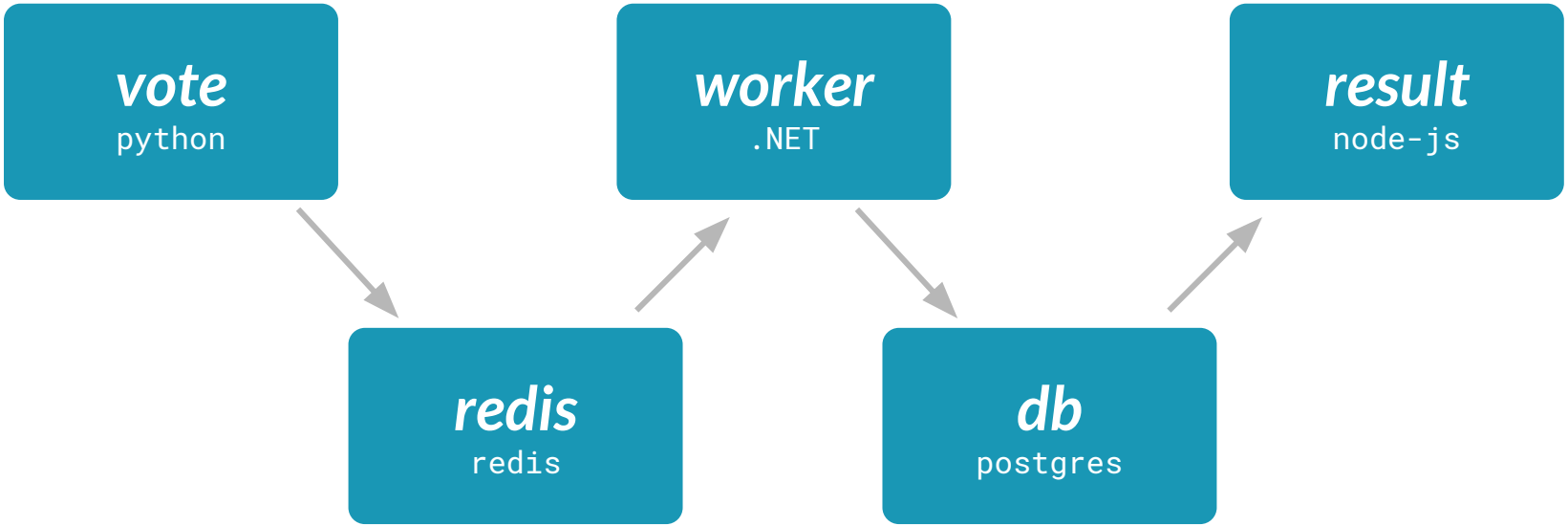
vote

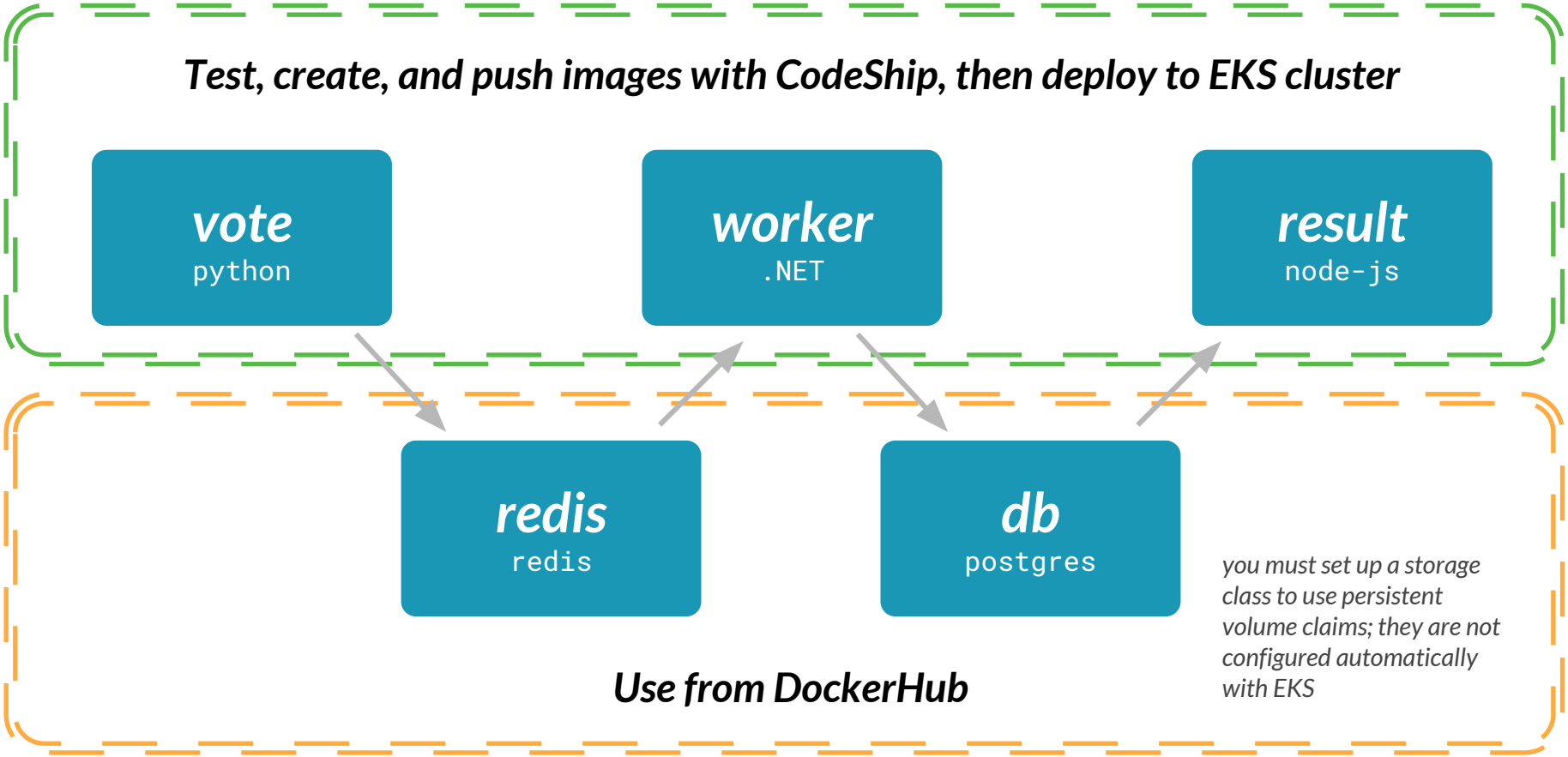


result

redis worker db

Service Architecture




Test, create, and push images with CodeShip, then deploy to EKS cluster
vote
python
worker
.NET
result
node-js
redis
redis
db
postgres
you must set up a storage class to use persistent volume claims; they are not configured automatically with EKS
Use from DockerHub

Switching Between Local Dev and EKS

see all available contexts

```
$ kubectl config get-contexts
CURRENT   NAME      CLUSTER      AUTHINFO   NAMESPACE
          aws       kubernetes   aws
```

add another config file to KUBECONFIG path

```
$ echo $KUBECONFIG
/Users/laura/.kube/config-demo
$ export KUBECONFIG=$KUBECONFIG:/Users/laura/.kube/config-docker4mac
```

new context has been added

```
$ kubectl config get-contexts
CURRENT   NAME                 CLUSTER                      AUTHINFO           NAMESPACE
*         aws                  kubernetes                   aws
          docker-for-desktop   docker-for-desktop-cluster   docker-for-desktop
```

...Or Just Update KUBECONFIG

see all available contexts

```
$ kubectl config get-contexts
CURRENT   NAME      CLUSTER      AUTHINFO   NAMESPACE
          aws       kubernetes   aws
```

update KUBECONFIG to only see one config file

```
$ echo $KUBECONFIG
/Users/laura/.kube/config-demo
$ export KUBECONFIG=/Users/laura/.kube/config-docker4mac
```

only one context!

```
$ kubectl config get-contexts
CURRENT   NAME                 CLUSTER                      AUTHINFO           NAMESPACE
          docker-for-desktop   docker-for-desktop-cluster   docker-for-desktop
```

Updating our Application

1. Make a change locally
2. Run tests locally
3. Push to GitHub & trigger a build on CodeShip
4. Build the updated images
5. Run tests against new code
6. Push new images to registry (Docker Hub)
7. Green build? Merge to master!
8. Use CodeShip to trigger a deployment on EKS
9. Finally see our changes in prod!

Updating our Application

1. Make a change locally
2. Run tests locally
3. Push to GitHub & trigger a build on CodeShip
4. Build the updated images
5. Run tests against new code
6. Push new images to registry (Docker Hub)
7. Green build? Merge to master!
8. Use CodeShip to trigger a deployment on EKS
9. Finally see our changes in prod!

Automate with CodeShip

Accessing your EKS Cluster from CodeShip

Prerequisites

- AWS account and credentials
- kubectl installed and configured locally
- The Jet CLI installed locally (bit.ly/codeship-jet-tool)

AWS access keys + kubeconfig allow you to access your EKS cluster from a CodeShip build.

EKS uses IAM credentials to authenticate to your cluster.

The `aws-iam-authenticator` was previously called `heptio-authenticator-aws`

Accessing your EKS Cluster from CodeShip

eks-env

```
AWS_ACCESS_KEY_ID=your_access_key_id
AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

Set up access to your cluster as described in the AWS EKS docs. Then flatten your kubeconfig and add it to your environment file. Use the Jet CLI and your project's AES key to encrypt the env file.

```
kubectl config current-context #make sure it's aws
kubectl config view --minify --flatten > kubeconfigdata
docker run --rm -it -v $(pwd):/files codeship/env-var-helper cp \
     kubeconfigdata:/root/.kube/config k8s-env
cat k8s-env >> eks-env
jet encrypt eks-env eks-env.encrypted
rm kubeconfigdata k8s-env eks-env #or add them to your .gitignore
```

Accessing your EKS Cluster from CodeShip

codeship-services.yml

```
[...]

kubectl:
  image: codeship/eks-kubectl
  encrypted_env_file: eks-env.encrypted
  volumes:
    - ./deploy:/deploy
```

This image has AWS-vendored kubectl, aws-iam-authenticator, and a helper script to pull the kubeconfig out of the encrypted environment variable and put it into /.kube/kubeconfig.

codeship-steps.yml

```
[...]

- name: eks_deployment
  service: kubectl
  tag: master
  command: ./deploy/eks-deployment.sh
```

This step is only run on builds from the master branch, and will run the EKS deploy script that is mounted into the container.

Deploying to EKS

Deploying to EKS

Managed, not magic

EKS is concerned with infrastructure, and doesn't replace existing deployment patterns

You still need to:

1. Package your application code in container images
2. Push the images to a registry for distribution
3. Issue commands to update your cluster
4. Deal with extra requirements like storage classes, etc

Good news: using EKS doesn't lock you in to ECR (AWS's image registry), though it may be slightly easier to use because of shared credentials and roles

Deploying to EKS

1. Build images with CodeShip and push to a registry using CodeShip's `push` step type.

codeship-steps.yml

```
- type: push
  service: worker
  name: push_worker_image
  image_name: rheinwein/examplevotingapp-worker
  image_tag: "{{.CommitID}}"
```

2. Use a deploy script to issue update commands against your EKS cluster.

codeship-steps.yml

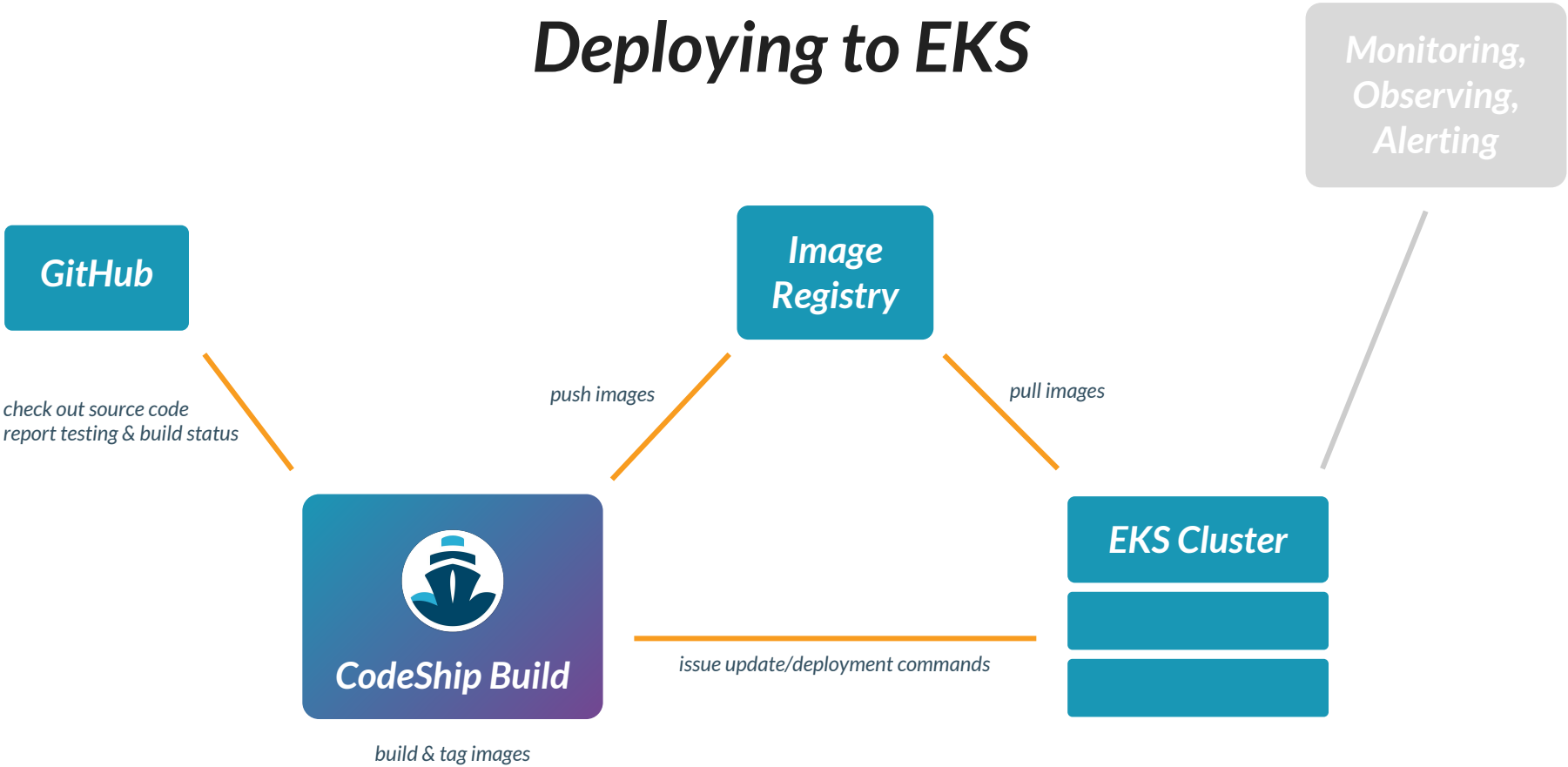
```
- name: eks_deployment
  service: kubectl
  tag: master
  command: ./deploy/eks-deployment.sh
```

Best Practice for Containerized Apps

Tag your images with versions or the commit SHA. Avoid pulling images using the `latest` tag.

Deploying to EKS

Deploying to EKS



CI/CD Tips and Best Practices

Encryption

CodeShip provides each project with an AES key to encrypt secrets. Using the CodeShip CLI jet, you can encrypt and decrypt environment variables.

Healthchecks

A running container only means that the process is running, not that the service is available. CodeShip respects the HEALTHCHECK attribute of services, and will wait until the service is available before trying to use it in a build.

Manual Approval

Want an extra set of eyes on changes before they're deployed, or want to restrict deployments to certain groups of people? With manual steps, you have more control over your CD process.

Useful Links

Sign up for CodeShip
https://codeship.com

AWS EKS Getting Started Guide
https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html

Download a local Kubernetes environment with Docker for Mac or Windows
https://www.docker.com/products/docker-desktop

Example-voting-app source code
https://github.com/rheinwein/example-voting-app

Download Deploying to Kubernetes Codeship eBook
https://resources.codeship.com/ebook/deploy-docker-kubernetes-codeship

Interested in learning more about DevOps best practices and use cases?

Join us for Jenkins World | DevOps World

San Francisco, California
September 16-19, 2018

Nice, France
October 22-25, 2018

Get 20% off with code **JWLTACHO**

Thank you!

Slides: bit.ly/eks-codeship