John David Luther Follow

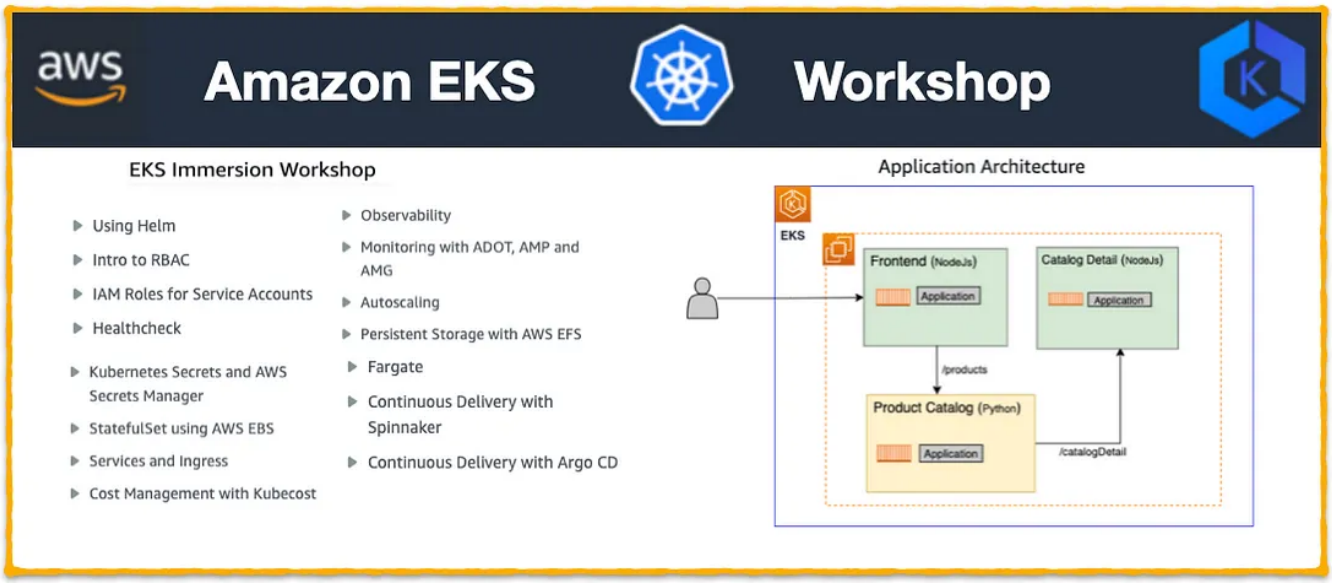
Oct 23, 2022 · 5 min read · ✦ · Listen

Working The Amazon EKS Immersion Workshop — Chapter 1 — Deploying A Microservices Application In A Kubernetes Cluster

A chapter by chapter journey from start to finish!

> "AWS Cookbook Secret — Taste the already baked workshops first and then bake a few more batches yourself!"
>
> Your Truly

Table of Contents



AWS Workshop Studio

Introduction

The best and the easiest way to get hands on with AWS is by practicing the tutorials, labs, and workshops. If you were to pursue that path, there is an unlimited supply of contents out there, coming from a variety of inspiring sources, including Amazon itself being the chief purveyor of these worthy goals.

My last two posts shown below simply scratched the rich surface of the resources, one in a general sense, and the other focused specifically on Amazon EKS.

12 AWS Essential Bookmarks

All in one place. Convenient access. Saves time. Boosts learning 100x!

awstip.com

Amazon Elastic Kubernetes Service (Amazon EKS) — The Only Resource Hub You Ever Need

16 must know resources to navigate, learn and master Amazon EKS!

awstip.com

One of the above two, the 16 must know resources to navigate, learn and master Amazon EKS!, picked the **EKS Immersion Workshop** as the top honoree. The reason being, it matched the strategy and execution plan that I had laid out. It is the best two-in-one workshop that teaches both Kubernetes and Amazon EKS by taking a step by step systematic approach.

Your mileage may vary but when I tried hands-on, even a well-organized workshop such as this posed some difficulties for me and I am sure they will not escape you either.

1. It takes a few hours to cut through the noises, get a hang of the overall workshop, build the cluster, do the necessary setup, etc. before you start rolling.
2. It takes days or even weeks if you are seriously trying out every detail as prescribed to complete the workshop.
3. Repeating the workshop, a must imperative, comes down to repeating the rigmaroles all over again. Unless you've preserved your previous edits and notes, oops!
4. Overall, all the above means huge inertia to get started. Even if you get started, there's a high likelihood of not finishing it. Even if you finish once, there's a low likelihood of ever repeat it. That's an unaffordable loss which I can help reverse.
5. **Ask Questions** — Most importantly, the workshops have no personal support but here if you have a question to clarify or you need a hand in your learning journey, post a comment, and I can help.

That's why I organized my own practice sessions into mini-chapters aligned with the workshop sections and it's written in a way that anyone can get started with any chapter in a matter of seconds by simply copy/pasting the commands, watching the output and internalizing the concepts. Then repeat as many times as you like with joy, not burden.

> *1. Personally, I feel it's the best way to working this, as well as all the other AWS workshops. Hope you like and take advantage of this chapter-by-chapter journey.*
>
> *2. You do need to supplement with parallel study of the workshops which contain a rich explanation of concepts. I grasp better after a few rounds of coding and hands-on practice of what's laid out here. Again YMMV!*

> *3. You do have full liberty to skip this and do the workshop fully on your own.*

Nuff said, let's code!

CH 1.1 — Setup Kubernetes Cluster on Amazon EKS

***Objective** — Prime the dev env with pre-reqs and create cluster using EKSCTL.*

We just want to stick to the stated objective here, which is to set up the dev environment that'd fire all the subsequent commands and create the Kubernetes cluster necessary for the rest of the workshop chapters, including the present one.

Since we're not participants of the workshop, it's perfectly fine to skip over the initial parts unless one wishes to read up for knowledge sake. The code below accomplishes the objective sufficiently. We'll just create the cluster, test it's up, running, accessible and proceed to deploying the application in the next section.

```
#!/usr/bin/bash

# Purpose: Working The Amazon EKS Immersion Workshop - CH1.1 - Setup Kubernetes Cluste
# Ref: https://catalog.workshops.aws/eks-immersionday/en-US/introduction/workshop-setu
# Blog Ref: https://medium.com/aws-tip/working-the-amazon-eks-immersion-workshop-chapt

#
# Author's NOTE
#  1. # are comment lines
#  2. Command output wherever helpful is shown inside {}
#

############
# STEP 01 - Setup tools, Connect and Configure AWS Account Connectivity
############

# Prerequisites- Tools
# The following tools are required to be functioning on the development terminal
#  1. AWS CLI - https://docs.aws.amazon.com/cli/latest/userguide/getting-started-insta
#  2. kubectl - https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html
#  3. eksctl - https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html
#  4. helm - https://helm.sh/docs/intro/install/

# Test to make sure all the CLI tools are functioning
aws --version
{
aws-cli/2.5.6 Python/3.9.11 Darwin/21.3.0 exe/x86_64 prompt/off
}

kubectl version --client=true --short
{
Client Version: v1.22.6-eks-7d68063
}

eksctl version
{
0.115.0
}

helm version --short
{
v3.8.2
}

# Configure AWS credentials for the account to run on
```

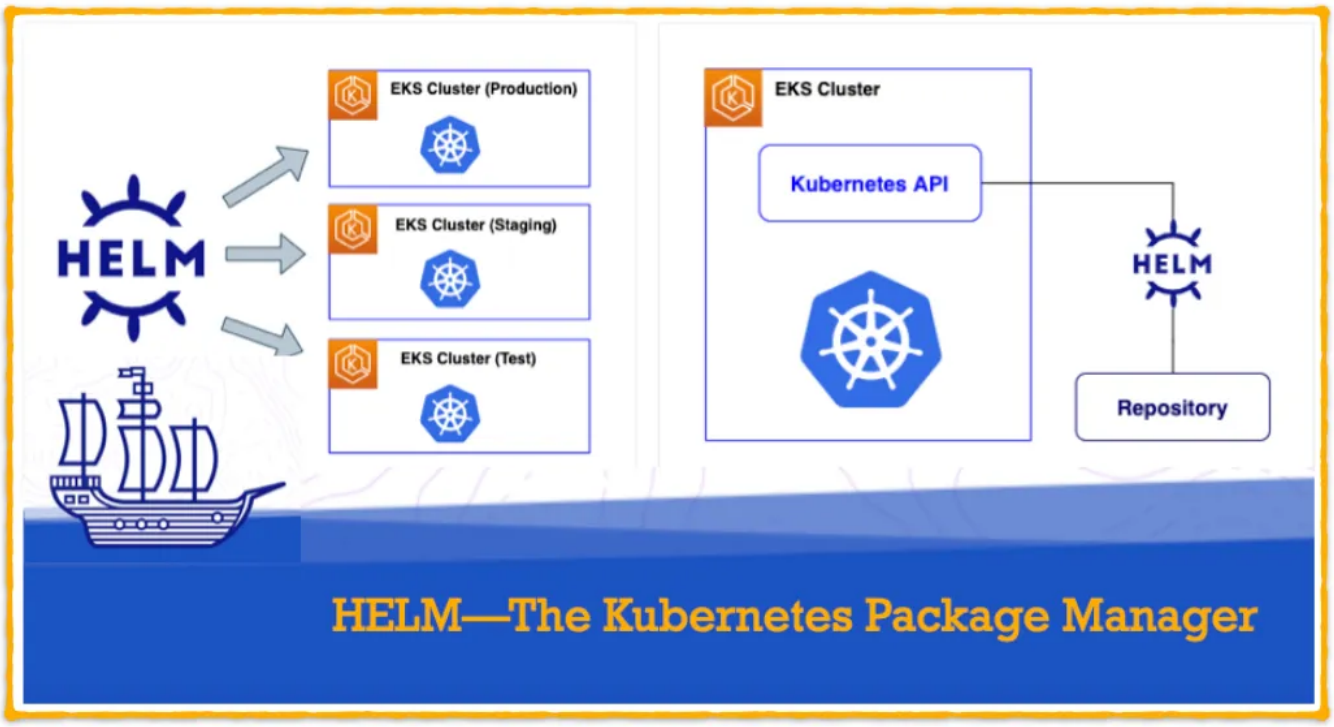
```
aws configure
{
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
}

# Test connectivity to AWS account
aws sts get-caller-identity
{
    "UserId": "AKIAIOSFODNN7EXAMPLE",
    "Account": "060066512345",
    "Arn": "arn:aws:iam::060066512345:user/cloud_user"
}


###########
# STEP 02 - Set ENV vars and env display utility for repeat convenience
###########

export WORKSHOP_HOME=~/amazon-eks-immersion/
export ACCOUNT_ID=$(aws sts get-caller-identity --output text --query 'Account')
export CLUSTER_NAME="eks-test-cluster"
export WORKSPACE_NAME="workshop"
export INSTANCE_TYPE="t3.medium"
export AWS_DEFAULT_REGION="us-east-1"
export AWS_REGION=$AWS_DEFAULT_REGION
export CLUSTER_ENDPOINT="TBD post cluster creation"


# Print env vars to see anytime, related to this book
function display_env() {
echo "WORKSHOP_HOME: [$WORKSHOP_HOME]"
echo "AWS_DEFAULT_REGION: [$AWS_DEFAULT_REGION]"
echo "AWS_REGION: [$AWS_REGION]"
echo "ACCOUNT_ID: [$ACCOUNT_ID]"
echo "CLUSTER_NAME: [$CLUSTER_NAME]"
echo "WORKSPACE_NAME: [$WORKSPACE_NAME]"
echo "INSTANCE_TYPE: [$INSTANCE_TYPE]"
echo "CLUSTER_ENDPOINT: [$CLUSTER_ENDPOINT]"
}

###########
# STEP 03 - Create Cluster
###########
```

```
""""""""""""

# Review and make sure env vars are set
display_env

# Get inside the main workship dir
mkdir -p $WORKSHOP_HOME
cd $WORKSHOP_HOME

# Create cluster using eksctl
eksctl create cluster -f - << EOF
---
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata:
  name: ${CLUSTER_NAME}
  region: ${AWS_DEFAULT_REGION}
managedNodeGroups:
  - name: ${CLUSTER_NAME}-ng
    instanceType: ${INSTANCE_TYPE}
    desiredCapacity: 3
    minSize: 2
    maxSize: 4
EOF
{
...
2022-10-22 17:39:09 [ℹ]  nodegroup "eks-test-cluster-ng" has 3 node(s)
2022-10-22 17:39:09 [ℹ]  node "ip-192-168-0-78.ec2.internal" is ready
2022-10-22 17:39:09 [ℹ]  node "ip-192-168-2-201.ec2.internal" is ready
2022-10-22 17:39:09 [ℹ]  node "ip-192-168-52-95.ec2.internal" is ready
2022-10-22 17:39:11 [ℹ]  kubectl command should work with "/Users/john.luther/.kube/cc
2022-10-22 17:39:11 [✔]  EKS cluster "eks-test-cluster" in "us-east-1" region is ready
}

export CLUSTER_ENDPOINT="$(aws eks describe-cluster --name ${CLUSTER_NAME} --query "cl
echo $CLUSTER_ENDPOINT
{
CLUSTER_ENDPOINT: [https://24596370E9B2B063E1FB06B864E704B6.gr7.us-east-1.eks.amazonaw
}

# Test to make sure the cluster is up, running and accessbile
kubectl get nodes
{
NAME                             STATUS   ROLES    AGE   VERSION
ip-192-168-0-78.ec2.internal     Ready    <none>   51m   v1.23.9-eks-ba74326
ip-192-168-2-201.ec2.internal    Ready    <none>   51m   v1.23.9-eks-ba74326
```

```
137   ip-192-168-52-95.ec2.internal   Ready    <none>   51m   v1.23.9-eks-ba74326
138   }
```

CH 1.2 — Deploy Microservices Application Using Helm

Objective *— Deploy Product Catalog Application to EKS cluster using Helm Chart.*



https://catalog.workshops.aws/eks-immersionday/en-US/helm

The workshop has a quick primer and few pointers to explain what Helm is. I'd recommend to proceeding from this page to the official Helm page for later reference.

What's most important as part of the objective here is to understand the microservices based product catalog application that we are about to deploy using the code below.

If all goes well, you should be able to access and interact with the application. More interesting aspects of the workshop will follow in the coming chapters.

The Product Catalog Application's final resource manifest file has all the parameters and resource specifications for your reference and deeper learning.

```
#!/usr/bin/bash

# Purpose: Working The Amazon EKS Immersion Workshop - CH 1.2 - Deploy Microservices Ap
# Ref: https://catalog.workshops.aws/eks-immersionday/en-US/helm
# Blog Ref: https://medium.com/aws-tip/working-the-amazon-eks-immersion-workshop-chapte
#
# Author's NOTE
#  1. # are comment lines
#  2. Command output wherever helpful is shown inside {}
#

# Set working location
cd $WORKSHOP_HOME

# Clone repo unless cloned already
git clone https://github.com/aws-containers/eks-app-mesh-polyglot-demo.git

cd eks-app-mesh-polyglot-demo

# Dry run.
# This step is optional. No need to repeat this run after the first run.
# NOTE - Dry run out produces the manifest that's worth taking a look.
helm install --debug --dry-run workshop ./workshop/helm-chart/

# Final run
helm install workshop ./workshop/helm-chart/
{
NAME: workshop
LAST DEPLOYED: Sat Oct 22 17:52:44 2022
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
     NOTE: It may take a few minutes for the LoadBalancer to be available.
           You can watch the status of by running 'kubectl get --namespace workshop svc
  export LB_NAME=$(kubectl get svc --namespace workshop frontend -o jsonpath="{.status.
  echo http://$LB_NAME:80
}

# List chart to confirm
helm list
{
NAME            NAMESPACE       REVISION        UPDATED
workshop        default         1               2022-10-22 17:52:44.746801 -0700 PDT
```

```
}

# Confirm pods and services are up
kubectl get pod,svc -n workshop -o wide
{
NAME                               READY   STATUS    RESTARTS   AGE     IP
pod/frontend-5ffb7ffc69-42hnz      1/1     Running   0          2m17s   192.168.51.0
pod/prodcatalog-5c47947fb7-4jg84   1/1     Running   0          2m17s   192.168.31.49
pod/proddetail-69d4b5fbdc-fjvz5    1/1     Running   0          2m17s   192.168.46.226

NAME                  TYPE           CLUSTER-IP      EXTERNAL-IP
service/frontend      LoadBalancer   10.100.90.2     a7c8fc2ca26e34965ab2fe73aa78d7df-1
service/prodcatalog   ClusterIP      10.100.247.9    <none>
service/proddetail    ClusterIP      10.100.12.144   <none>
}

# Get the Loadbalancer url above or with command below anytime to access the applicatio
# It may take a few minutes for the Load Balancer to be up
export LB_NAME=$(kubectl get svc frontend -n workshop -o jsonpath="{.status.loadBalance
echo $LB_NAME
{
http://a7c8fc2ca26e34965ab2fe73aa78d7df-1220770866.us-east-1.elb.amazonaws.com:80
}

# http://a7c8fc2ca26e34965ab2fe73aa78d7df-1220770866.us-east-1.elb.amazonaws.com should
# Test the ADD functionality by adding a product id and product name
```

amazon-eks-immersion-ch1.2.sh hosted with ❤ by GitHub view raw

Conclusion

You have in your hands the first chapter of the chapter-by-chapter journey from start to finish of ***The Amazon EKS Immersion Workshop***.

The code provided in this chapter helps to achieve two main objectives. The rest of the workshop will depend on this foundation to demonstrate many more powerful features and capabilities of Kubernetes and Amazon EKS.

First, setup the dev terminal and create a Kubernetes cluster using **`eksctl`**, the official CLI for Amazon EKS. The code here can be used to create a cluster in a matter of seconds to work on this workshop or any other Kubernetes work.

Second, deploy a microservices based web application using helm, the package

manager for Kubernetes.

The code should work as is, "*out of the box*", by simply copy/pasting each command in the order specified. Feel free to repeat it as many times as necessary to feel comfortable and internalize the concepts and the commands.

See you soon in the next chapter.

Working The Amazon EKS Immersion Workshop — Chapter 2— Kubernetes RBAC (Role-Based Access Control)

AWS Kubernetes Helm Microservices Cloud Computing

awstip.com

Please follow to stay in touch, track and be the first one to get notified of my future writings on clouds, containers, Kubernetes, MLOps and AWS. Please bookmark This Code Works! — Channel Home Page below for easy access to all posts in a single place. Enjoy!