PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

www.zebrax.id

#PHPID Online Learning

Laravel Kubernetes (EKS) Deployment Using Terraform & HelmChart

Selasa, 02 November 2021 - 19.00 WIB

Dwi Fahni Denni *as Speaker*

DevOps Lead *ZebraX*

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

www.zebrax.id

Dwi Fahni Denni

https://www.linkedin.com/in/dfdenni

https://github.com/zeroc0d3
https://github.com/ZXDfdenni

About Speaker

Coffee Addict

- **Carrier Path**
  - Software Engineer (2009-2015)
  - System Administrator (2015-2017)
  - DevOps Engineer (2017 - now)
  - DevOps Engineer Lead @ ZebraX (2021 - now)
  - AWS Community Builder, Container Categories (2021)
- **Education**
  - Electronics Engineer
    Institut Teknologi Nasional (ITENAS), Bandung
- **Hobby**
  - Gaming (Minecraft & Dota 2)
  - Sport (Archery)

ZEBRAX

ABOUT US

ZebraX is a **digital transformation company** that combines world class digital technology expertise with in-field experience and local know how. It is an industrial tech company established by Indika Energy, Indonesia's leading integrated energy company, with robust partnership network in digital connectivity, automation, and deep learning analytics.

OUR MISSION

By **bridging the gap between data and business impact,** ZebraX aims to help clients effectively harness digital technology to **bring their operations to the next level.**

Our Key Capabilities

Digital core capabilities to provide the best result in enabling Industry 4.0 journey

Data Science and Industrial IoT

Delivering enterprise-scale machine learning and advanced analytics applications and enabling automated data capture using specifically designed industrial grade IoT devices.

Digital Strategy and Business Consulting

Enabling Industry 4.0 by identifying business value through value-at-stake analysis, digital implementation road-mapping, and mobilizing suitable IoT strategy.

Product & Technology Development

Building superior digital products by top-tier tech architect, developers, and UI/UX designers with both vast and deep knowledge in newest technology platforms.

ZEBRA

OUR SOLUTIONS Advanced Analytics

Advanced Analytics Services

DATA ACQUISITION & PROCESSING

- Inspection data
- Legacy sensors (PLC, DCS)
- IoT sensors
- Maintenance data

ANALYTICS & MACHINE LEARNING

- Modelling
- Historical data analysis
- Domain expert

VISUALIZATION & DECISION MAKING

- Predictions & simulations
- Data visualization
- Real-time data update

"Deliver data-driven, intelligent insight for performance optimization"

ENABLING INDUSTRY 4.0 FOR YOUR BUSINESS

ZEBRA

"Together, we can enable Industry 4.0 for your business with zero risk. Let's discover on how ZebraX can make unintuitive connections for your business."

Get Connected with Us!

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

www.zebrax.id

Contents

Requirements

- Authorize: AWS Console Access & IAM Policy
- Container Registry: ECR / DockerHub
- CLI: aws, eksctl, docker, docker-compose, terraform, helm, kubectl, kubectx
- Linux: Basic Linux Command & Networking
- Deploy: Workflow Deployment
- Container: Basic Container (Docker) & Orchestration (K8S)

Excludes

- CI/CD Tools: AWS CodePipeline, Jenkins, GitHub Action, GitLab CI/CD, Bitbucket Pipeline, Spinnaker, etc
- Monitoring Kubernetes Metrics & Logs: Datadog, Prometheus, Grafana, Elasticsearch, etc
- Service Mesh: AWS AppMesh, Traefik, Envoy, Istio, etc
- Secret Config: AWS KMS, AWS System Manager Parameter Store, Hashicorp Vault, etc

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

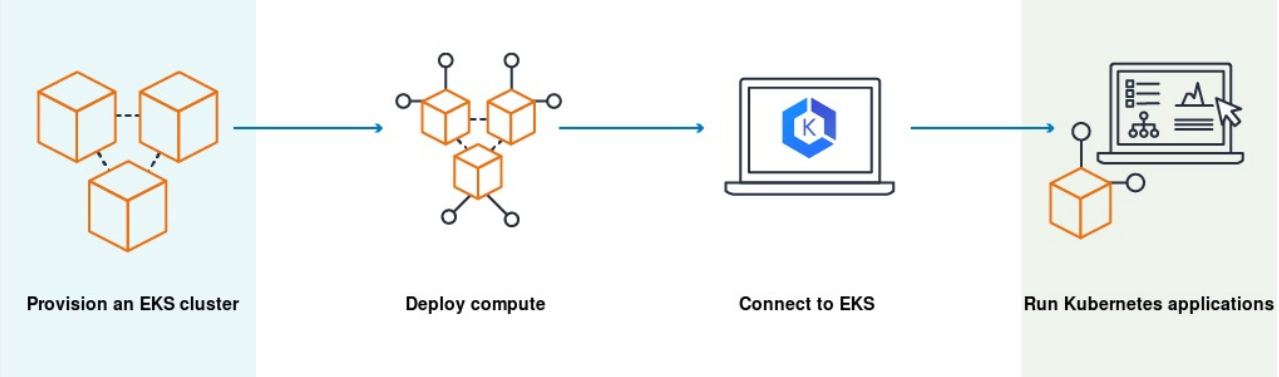
www.zebrax.id

Contents

Introduction Amazon EKS

- How Does Amazon EKS Work
- Amazon EKS Components
  - Amazon EKS Clusters
  - Amazon EKS Nodes
  - Amazon EKS Networking
- Amazon EKS Storage

How Does Amazon EKS Work



- EKS Cluster composed with control plane and worker nodes.
- Each cluster runs its own, fully managed Virtual Private Cloud (VPC).
- Control planes running with 3 nodes, each nodes place in different AZ for AWS high availability.
- Incoming traffic directed to the Kubernetes (K8S) API passed through the AWS Network Load Balancer (NLB).
- Worker nodes run on Amazon EC2 instances located in VPC (self-manged).
- Deployment can be one cluster for each environment application, or define IAM security policies and K8S namespaces to deploy one cluster for multi applications.

Amazon EKS

Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Amazon EKS is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.
- It typically runs in the Amazon public cloud, but can also be deployed on premises.
- The Kubernetes management infrastructure of Amazon EKS runs across multiple Availability Zones (AZ).
- AWS EKS is certified Kubernetes-conformant, which means you can integrate EKS with your existing tools.
- **References:**
  - What is Amazon EKS? - Amazon EKS
  - AWS EKS Architecture: Clusters, Nodes, and Networks (netapp.com)
  - Amazon EKS networking - Amazon EKS

Amazon EKS Components -1-

- Amazon EKS Clusters
  - EKS Control Planes
    - Dedicated EC2 Instances in Amazon-Managed AWS.
    - Provide API endpoint for your applications.
    - Running single-tenant mode (allows organizations of any size to deploy the gateway as an exterior, or Internet-facing edge virtual private network (VPN) and DirectAccess server).
    - Responsibility for controlling Kubernetes (K8S) master nodes: API Server & etcd.
  - EKS Nodes
    - Running on EC2 instances in your organization's AWS account.
    - Use API endpoint to connect the control plane, via certificate file.
    - Unique certificate used in each cluster.

EKS Cluster

Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Amazon EKS is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.
- It typically runs in the Amazon public cloud, but can also be deployed on premises.
- The Kubernetes management infrastructure of Amazon EKS runs across multiple Availability Zones (AZ).
- AWS EKS is certified Kubernetes-conformant, which means you can integrate EKS with your existing tools.
- **References:**
  - What is Amazon EKS? - Amazon EKS
  - AWS EKS Architecture: Clusters, Nodes, and Networks (netapp.com)
  - Amazon EKS networking - Amazon EKS

Amazon EKS Components –2.1–

- Amazon EKS Nodes
  - Self-Managed Nodes
    - Amazon EC2 Instance that K8S pods can be scheduled on.
    - Pods connect to EKS cluster's API endpoint.
    - Nodes are organized into node groups.
    - All the EC2 instances in node group must have the same:
      - Amazon Instance Type.
      - Amazon Machine Image (AMI).
      - IAM Role.
  - Managed Node Groups
    - Running on EC2 instances in your organization's AWS account.
    - Use API endpoint to connect the control plane, via certificate file.
    - Unique certificate used in each cluster.
  - Amazon Fargate
    - AWS Fargate is a serverless compute engine for containers.
    - Serverless container service, to run worker nodes without the underlying infrastructure.

EKS Nodes

Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Amazon EKS is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.
- It typically runs in the Amazon public cloud, but can also be deployed on premises.
- The Kubernetes management infrastructure of Amazon EKS runs across multiple Availability Zones (AZ).
- AWS EKS is certified Kubernetes-conformant, which means you can integrate EKS with your existing tools.
- **References:**
  - What is Amazon EKS? - Amazon EKS
  - AWS EKS Architecture: Clusters, Nodes, and Networks (netapp.com)
  - Amazon EKS networking - Amazon EKS

Amazon EKS Components –2.2–

- Amazon Fargate
  - AWS Fargate is a serverless compute engine for containers.
  - Serverless container service, to run worker nodes without the underlying infrastructure.
  - With Fargate, you can focus on building and operating your applications whether you are running it with ECS or EKS. You only interact with and pay for your containers, and you avoid the operational overhead of scaling, patching, securing, and managing servers. Fargate ensures that the infrastructure your containers run on is always up-to-date with the required patches.
  - With Fargate, you get out-of-box observability through built-in integrations with other AWS services including Amazon CloudWatch Container Insights.
  - Fargate allows you to gather metrics and logs for monitoring your applications through an extensive selection of third party tools with open interfaces.

EKS Nodes

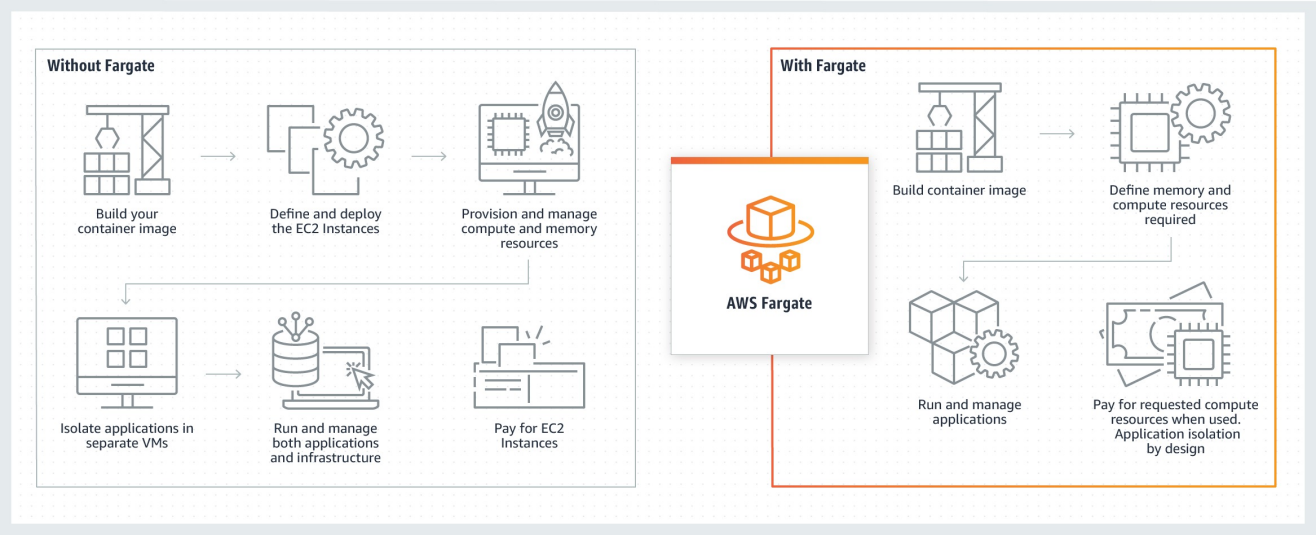
Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Amazon EKS is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.
- It typically runs in the Amazon public cloud, but can also be deployed on premises.
- The Kubernetes management infrastructure of Amazon EKS runs across multiple Availability Zones (AZ).
- AWS EKS is certified Kubernetes-conformant, which means you can integrate EKS with your existing tools.
- **References:**
  - What is Amazon EKS? - Amazon EKS
  - AWS EKS Architecture: Clusters, Nodes, and Networks (netapp.com)
  - Amazon EKS networking - Amazon EKS

Amazon EKS Components –2.3–

- Amazon Fargate
  - Fargate allows you to gather metrics and logs for monitoring your applications through an extensive selection of third party tools with open interfaces.



EKS Nodes

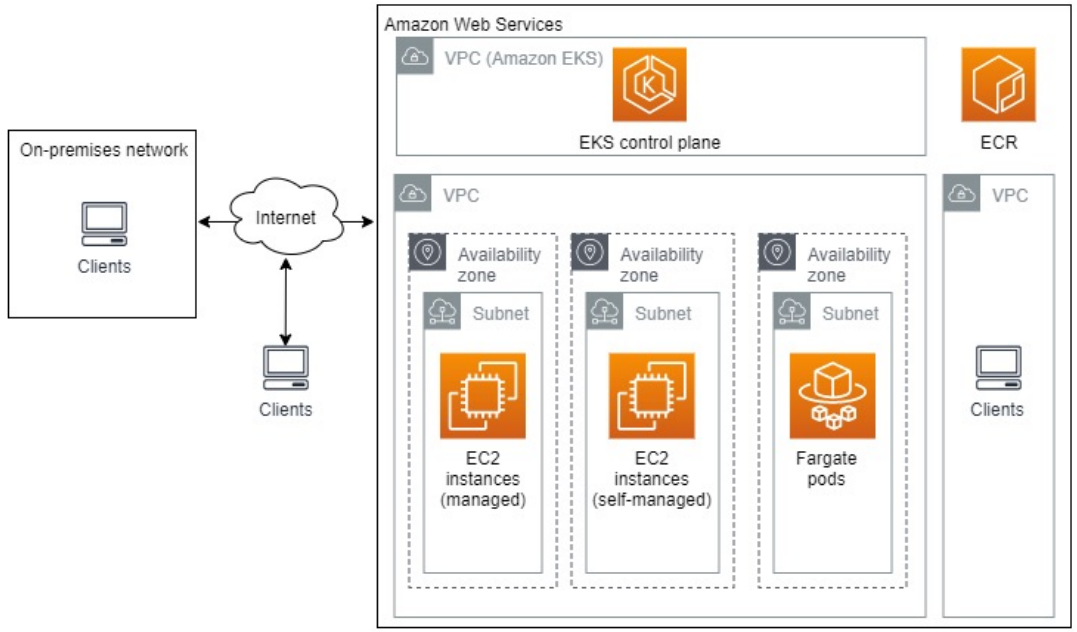
Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Amazon EKS is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.
- It typically runs in the Amazon public cloud, but can also be deployed on premises.
- The Kubernetes management infrastructure of Amazon EKS runs across multiple Availability Zones (AZ).
- AWS EKS is certified Kubernetes-conformant, which means you can integrate EKS with your existing tools.
- **References:**
  - What is Amazon EKS? - Amazon EKS
  - AWS EKS Architecture: Clusters, Nodes, and Networks (netapp.com)
  - Amazon EKS networking - Amazon EKS

Amazon EKS Components –3–

- Amazon EKS Networking



- EKS control plane runs in Amazon-manged VPC.
- Network interface use by EC2 & Fargate to connect EKS control plane.
- EKS expose a public endpoint as default.

EKS Networking

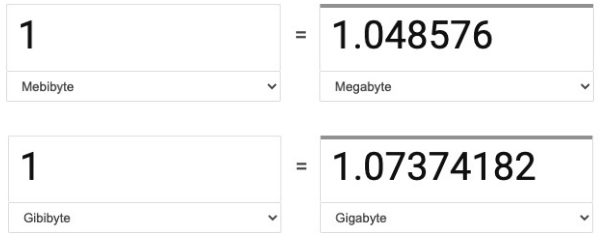
Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Amazon EKS is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.
- It typically runs in the Amazon public cloud, but can also be deployed on premises.
- The Kubernetes management infrastructure of Amazon EKS runs across multiple Availability Zones (AZ).
- AWS EKS is certified Kubernetes-conformant, which means you can integrate EKS with your existing tools.
- **References:**
  - What is Amazon EKS? - Amazon EKS
  - AWS EKS Architecture: Clusters, Nodes, and Networks (netapp.com)
  - Amazon EKS networking - Amazon EKS

Amazon EKS Storages


```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: slow
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```



- `type`: `io1`, `gp2`, `sc1`, `st1`. See AWS docs for details. Default: `gp2`.
- `iopsPerGB`: only for `io1` volumes. I/O operations per second per GiB. AWS volume plugin multiplies this with size of requested volume to compute IOPS of the volume and caps it at 20 000 IOPS (maximum supported by AWS, see AWS docs. A string is expected here, i.e. "10", not 10.
- `fsType`: `fsType` that is supported by kubernetes. Default: "ext4".
- `encrypted`: denotes whether the EBS volume should be encrypted or not. Valid values are "true" or "false". A string is expected here, i.e. "true", not true.
- `kmsKeyId`: optional. The full Amazon Resource Name of the key to use when encrypting the volume. If none is supplied but encrypted is true, a key is generated by AWS. See AWS docs for valid ARN value.

EKS Storages

Amazon EKS

Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that you can use to run Kubernetes on AWS without needing to install, operate, and maintain your own Kubernetes control plane or nodes.

- Amazon EKS is a managed AWS Kubernetes service that scales, manages, and deploys containerized applications.
- It typically runs in the Amazon public cloud, but can also be deployed on premises.
- The Kubernetes management infrastructure of Amazon EKS runs across multiple Availability Zones (AZ).
- AWS EKS is certified Kubernetes-conformant, which means you can integrate EKS with your existing tools.
- **References:**
  - What is Amazon EKS? - Amazon EKS
  - AWS EKS Architecture: Clusters, Nodes, and Networks (netapp.com)
  - Amazon EKS networking - Amazon EKS

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

www.zebrax.id

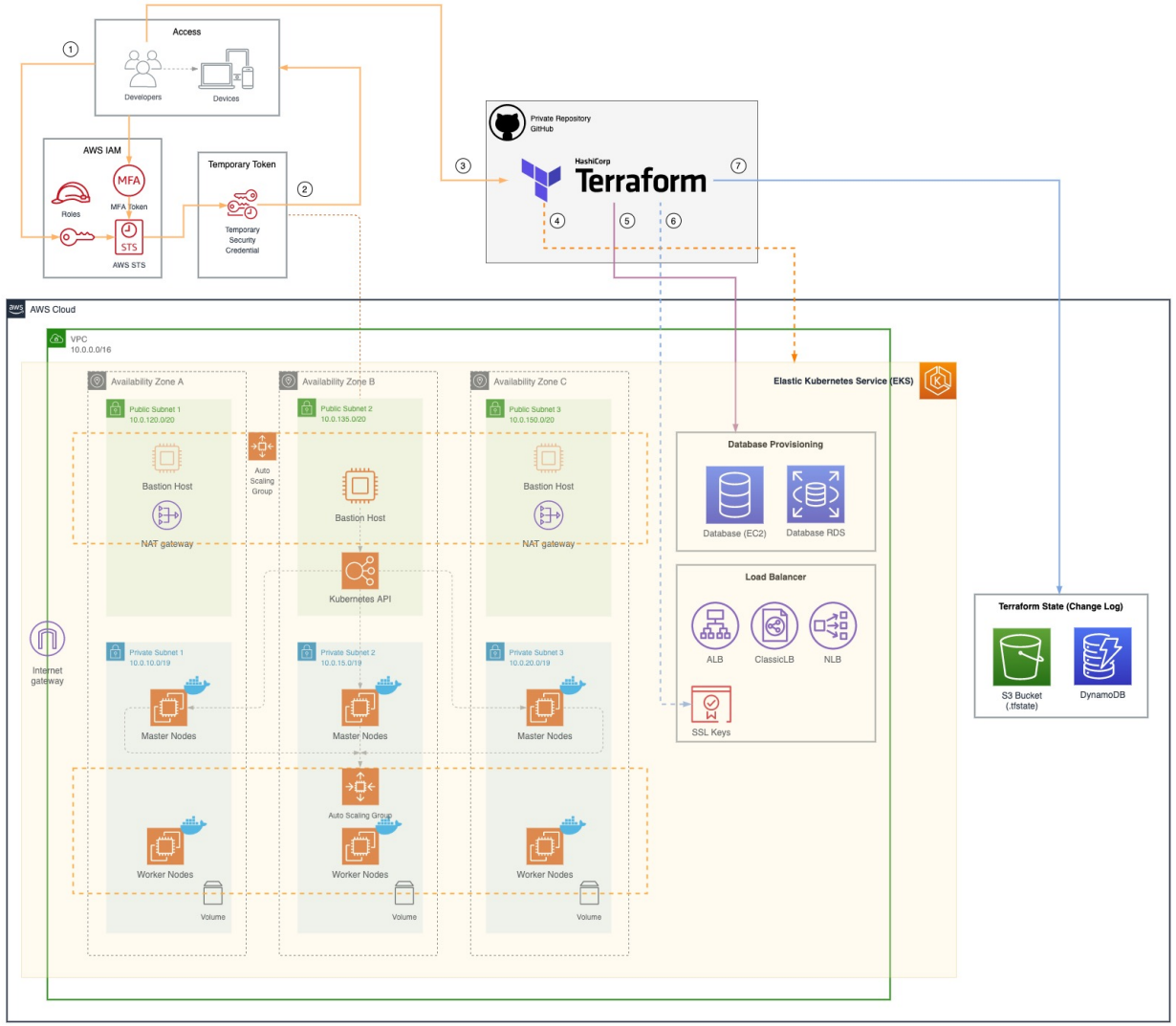
Contents

Terraform EKS Provisioning

- Introduction Terraform

  Ref: Webminar AWS Community Builders (@luthfianandra)
- Terraform Module for AWS EKS Provider
- Setup IAM Policy EKS for Terraform Authorization
- Execute EKS Provisioning
- Setup Authorization Cluster (mapRoles, mapUser)
- Setup Namespace, Service Account & Roles (Role, RoleBinding)
- Setup EKS Ingress Controller (Nginx & ALB)
- Setup EKS DNS in Route53

Architecture Terraform Provisioning



Architecture

Terraform CLI

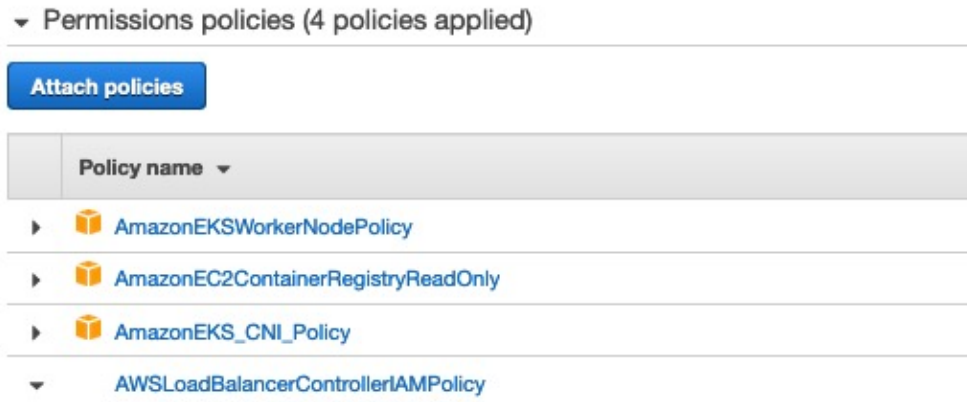
Terraform is an open-source infrastructure as code software tool that provides a consistent CLI workflow to manage hundreds of cloud services.

- Initialize Terraform
- Download Terraform modules from AWS provider with some of resources includes:
  - EC2
  - S3
  - EKS
  - Network (IGW, NAT)
- Create configuration provisioning inside Terraform script
- Execute Terraform Plan to review infrastructure changes
- Execute Terraform Apply to run infrastructure provisioning

Setup IAM Policy EKS for Terraform Authorization -1-

- IAM Policy for Cluster

- IAM Policy for Nodes



IAM Policy

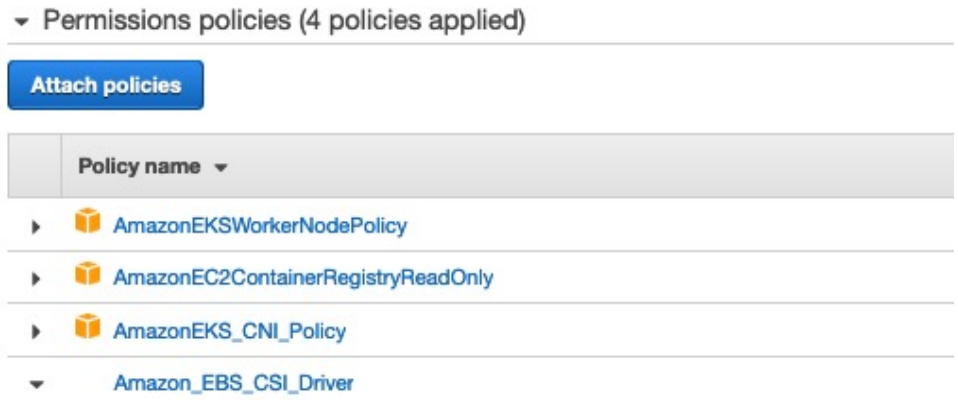
IAM Policy Terraform

IAM Policy will be use as EKS authorization to create, modified or delete/destroy EKS Cluster

- Terraform will add / update IAM Policy for EKS Cluster after this cluster successfully provisioned
- Do not manually edit / update the IAM Policy that already deployed by Terraform

Setup IAM Policy EKS for Terraform Authorization -2-

- IAM Policy for Node Groups



IAM Policy

IAM Policy Terraform

IAM Policy will be use as EKS authorization to create, modified or delete/destroy EKS Cluster

- Terraform will add / update IAM Policy for EKS Cluster after this cluster successfully provisioned
- Do not manually edit / update the IAM Policy that already deployed by Terraform

Execute EKS Provisioning

- Initialize Terraform (`terraform init`)

```
└─ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v3.63.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

- Validate Terraform (`terraform validate`)

```
└─ terraform validate
Success! The configuration is valid.
```

Terraform

Terraform CLI

Command for this execution: **`terraform validate`**, **`terraform plan`**, **`terraform apply`**, **`terraform destroy`**

- Make sure all values inside terraform script have been filled.
- Use: `` `terraform validate` `` for checking / validate all configuration inside terraform script
- Use: `` `terraform plan` `` for review changes of infrastructure
- Use: `` `terraform apply` `` for execute changes infrastructure
- Use: `` `terraform destroy` `` for delete / cleanup all changes infrastructure from terraform configuration
- Terraform tested environment version `` `1.0.9` ``

Execute EKS Provisioning

- Review Terraform (`terraform plan`) -1-

```
└─ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with
  + create

Terraform will perform the following actions:

  # aws_eip.nat will be created
  + resource "aws_eip" "nat" {
      + allocation_id        = (known after apply)
      + association_id       = (known after apply)
      + carrier_ip           = (known after apply)
      + customer_owned_ip    = (known after apply)
      + domain               = (known after apply)
      + id                   = (known after apply)
      + instance             = (known after apply)
      + network_border_group = (known after apply)
      + network_interface    = (known after apply)
      + private_dns          = (known after apply)
      + private_ip           = (known after apply)
      + public_dns           = (known after apply)
      + public_ip            = (known after apply)
      + public_ipv4_pool     = (known after apply)
      + tags_all             = (known after apply)
      + vpc                  = true
    }

  # aws_internet_gateway.igw will be created
  + resource "aws_internet_gateway" "igw" {
```

Terraform

Terraform CLI

Command for this execution: **`terraform validate`**, **`terraform plan`**, **`terraform apply`**, **`terraform destroy`**

- Make sure all values inside terraform script have been filled.
- Use: `` `terraform validate` `` for checking / validate all configuration inside terraform script
- Use: `` `terraform plan` `` for review changes of infrastructure
- Use: `` `terraform apply` `` for execute changes infrastructure
- Use: `` `terraform destroy` `` for delete / cleanup all changes infrastructure from terraform configuration
- Terraform tested environment version `1.0.9`

Execute EKS Provisioning

- Review Terraform (`terraform plan`) -2-

```
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostnames             = true
      + enable_dns_support               = true
      + id                               = (known after apply)
      + instance_tenancy                 = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                         = (known after apply)
      + tags                             = {
          + "Department"      = "DEVOPS"
          + "DepartmentGroup" = "DEV-DEVOPS"
          + "Environment"     = "DEV"
          + "Name"            = "devopscorner_vpc"
          + "ResourceGroup"   = "DEV-VPC"
        }
      + tags_all                         = {
          + "Department"      = "DEVOPS"
          + "DepartmentGroup" = "DEV-DEVOPS"
          + "Environment"     = "DEV"
          + "Name"            = "devopscorner_vpc"
          + "ResourceGroup"   = "DEV-VPC"
        }
    }

Plan: 15 to add, 0 to change, 0 to destroy.
```

Terraform

Terraform CLI

Command for this execution: **`terraform validate`**, **`terraform plan`**, **`terraform apply`**, **`terraform destroy`**

- Make sure all values inside terraform script have been filled.
- Use: `` `terraform validate` `` for checking / validate all configuration inside terraform script
- Use: `` `terraform plan` `` for review changes of infrastructure
- Use: `` `terraform apply` `` for execute changes infrastructure
- Use: `` `terraform destroy` `` for delete / cleanup all changes infrastructure from terraform configuration
- Terraform tested environment version `1.0.9`

Execute EKS Provisioning

- Provisioning Terraform (`terraform apply`)

```
aws_eks_node_group.devops_poc: Still creating... [10s elapsed]
aws_eks_node_group.devops_poc: Still creating... [20s elapsed]
aws_eks_node_group.devops_poc: Still creating... [30s elapsed]
aws_eks_node_group.devops_poc: Still creating... [40s elapsed]
aws_eks_node_group.devops_poc: Still creating... [50s elapsed]
aws_eks_node_group.devops_poc: Still creating... [1m0s elapsed]
aws_eks_node_group.devops_poc: Still creating... [1m10s elapsed]
aws_eks_node_group.devops_poc: Still creating... [1m20s elapsed]
aws_eks_node_group.devops_poc: Still creating... [1m30s elapsed]
aws_eks_node_group.devops_poc: Still creating... [1m40s elapsed]
aws_eks_node_group.devops_poc: Still creating... [1m50s elapsed]
aws_eks_node_group.devops_poc: Still creating... [2m0s elapsed]
aws_eks_node_group.devops_poc: Still creating... [2m10s elapsed]
aws_eks_node_group.devops_poc: Still creating... [2m20s elapsed]
aws_eks_node_group.devops_poc: Still creating... [2m30s elapsed]
aws_eks_node_group.devops_poc: Still creating... [2m40s elapsed]
aws_eks_node_group.devops_poc: Still creating... [2m50s elapsed]
aws_eks_node_group.devops_poc: Still creating... [3m0s elapsed]
aws_eks_node_group.devops_poc: Still creating... [3m10s elapsed]
aws_eks_node_group.devops_poc: Still creating... [3m20s elapsed]
aws_eks_node_group.devops_poc: Still creating... [3m30s elapsed]
aws_eks_node_group.devops_poc: Still creating... [3m40s elapsed]
aws_eks_node_group.devops_poc: Still creating... [3m50s elapsed]
aws_eks_node_group.devops_poc: Still creating... [4m0s elapsed]
aws_eks_node_group.devops_poc: Creation complete after 4m6s [id=devopscorner_poc:devopscorner-staging]

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.
```

Terraform

Terraform CLI

Command for this execution: **`terraform validate`**, **`terraform plan`**, **`terraform apply`**, **`terraform destroy`**

- Make sure all values inside terraform script have been filled.
- Use: `` `terraform validate` `` for checking / validate all configuration inside terraform script
- Use: `` `terraform plan` `` for review changes of infrastructure
- Use: `` `terraform apply` `` for execute changes infrastructure
- Use: `` `terraform destroy` `` for delete / cleanup all changes infrastructure from terraform configuration
- Terraform tested environment version `1.0.9`

Delete EKS Cluster

- Cleanup Environment (`terraform destroy`)

```
aws_iam_role_policy_attachment.AmazonEC2ContainerRegistryReadOnly: Destruction complete after 1s
aws_iam_role_policy_attachment.AmazonEKS_CNI_Policy: Destruction complete after 1s
aws_iam_role_policy_attachment.AmazonEKSServicePolicy: Destruction complete after 1s
aws_iam_role_policy_attachment.AmazonEKSWorkerNodePolicy: Destruction complete after 1s
aws_iam_role.eks_nodes: Destroying... [id=eks-role-devopscorner_poc-nodes]
aws_iam_role_policy_attachment.AmazonEKSClusterPolicy: Destruction complete after 1s
aws_iam_openid_connect_provider.cluster: Destruction complete after 2s
aws_eks_cluster.aws_eks: Destroying... [id=devopscorner_poc]
aws_iam_role.eks_nodes: Destruction complete after 6s
aws_eks_cluster.aws_eks: Still destroying... [id=devopscorner_poc, 10s elapsed]
aws_eks_cluster.aws_eks: Still destroying... [id=devopscorner_poc, 20s elapsed]
aws_eks_cluster.aws_eks: Still destroying... [id=devopscorner_poc, 30s elapsed]
aws_eks_cluster.aws_eks: Still destroying... [id=devopscorner_poc, 40s elapsed]
aws_eks_cluster.aws_eks: Still destroying... [id=devopscorner_poc, 50s elapsed]
aws_eks_cluster.aws_eks: Destruction complete after 55s
aws_iam_role.eks_cluster: Destroying... [id=eks-role-devopscorner_staging-cluster]
aws_iam_role.eks_cluster: Destruction complete after 5s

Destroy complete! Resources: 9 destroyed.
```

Terraform

Terraform CLI

Command for this execution: **`terraform validate`**, **`terraform plan`**, **`terraform apply`**, **`terraform destroy`**

- Make sure all values inside terraform script have been filled.
- Use: `` `terraform validate` `` for checking / validate all configuration inside terraform script
- Use: `` `terraform plan` `` for review changes of infrastructure
- Use: `` `terraform apply` `` for execute changes infrastructure
- Use: `` `terraform destroy` `` for delete / cleanup all changes infrastructure from terraform configuration
- Terraform tested environment version `` `1.0.9` ``

Setup Authorization EKS Cluster (mapRoles, mapUsers)

- ConfigMap for `aws-auth` in `kube-system` namespace

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - groups:
      - system:bootstrappers
      - system:nodes
      rolearn: arn:aws:iam::[AWS_ACCOUNT]:role/[EKS_ROLE_NAME]
      username: system:node:{{EC2PrivateDNSName}}
    - groups:
      - engineering:ops
      rolearn: arn:aws:iam::[AWS_ACCOUNT]:user/[SOME_USERNAME]
      username: [SOME_USERNAME]
  mapUsers: |
    - userarn: arn:aws:iam::[AWS_ACCOUNT]:user/[YOUR_EMAIL_ACCOUNT]
    username: [YOUR_EMAIL_ACCOUNT]
    groups:
      - system:masters
```

Authorization

Authorization EKS Cluster

Setup authorization EKS Cluster for ARN Role and/or User ARN

- Edit ConfigMap for Authorization Role and/or User ARN from `aws-auth`
  - `kubectl edit configmap -n kube-system aws-auth`

Setup Namespace, ServiceAccont, Roles (Role, RoleBinding) -1-

- Create Namespace

```
kubectl create namespace laravel-app
```

- Create Service Account

```
kubectl create serviceaccount laravel-svcaccount
```

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: laravel-svcaccount
  namespace: laravel-app
EOF
```

- Create Role

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ops-laravel-access
  namespace: laravel-app
rules:
- apiGroups: ["*"]
  resources: ["*"]
  verbs: ["patch", "get", "list", "watch", "create", "update", "delete"]
- apiGroups: ["autoscaling"]
  resources: ["horizontalpodautoscallers"]
  verbs: ["create", "delete", "patch", "update", "get", "watch", "list"]
```

RBAC

Authorization Service

Setup EKS Cluster RBAC (Role Access Base Controller) for specific namespace, service account and roles

- Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within your organization.
- *Namespaces* provides a mechanism for isolating groups of resources within a single cluster. Names of resources need to be unique within a namespace, but not across namespaces. Namespace-based scoping is applicable only for namespaced objects *(e.g. Deployments, Services, etc)* and not for cluster-wide objects *(e.g. StorageClass, Nodes, PersistentVolumes, etc).*
- A service account provides an identity for processes that run in a Pod.

Setup Namespace, ServiceAccont, Roles (Role, RoleBinding) -2-

- Create RoleBinding

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ops-laravel-access-binding
  namespace: laravel-app
subjects:
  - kind: Group
    name: engineering:ops
    apiGroup: rbac.authorization.k8s.io
  - kind: ServiceAccount
    name: laravel-svcaccount
roleRef:
  kind: Role
  name: ops-laravel-access
  apiGroup: rbac.authorization.k8s.io
```

RBAC

Authorization Service

Setup EKS Cluster RBAC (Role Access Base Controller) for specific namespace, service account and roles

- A role binding grants the permissions defined in a role to a user or set of users. It holds a list of *subjects* (users, groups, or service accounts), and a reference to the role being granted. A RoleBinding grants permissions within a specific namespace whereas a ClusterRoleBinding grants that access cluster-wide.
- A RoleBinding may reference any Role in the same namespace. Alternatively, a RoleBinding can reference a ClusterRole and bind that ClusterRole to the namespace of the RoleBinding. If you want to bind a ClusterRole to all the namespaces in your cluster, you use a ClusterRoleBinding.

Setup EKS Ingress Controller (NGINX, ALB) -1.1-

- Why we use Ingress NGINX Controller ?
  - There's no default port exposed in K8S using port 80 (http) & 443 (https).
  - There's no SSL Termination that will "force" redirect from http to https.
  - As mandatory services while you're not using service mesh / service discovery (eg: Traefik / Istio)

- Deploy Ingress NGINX Controller

```
## References: https://kubernetes.github.io/ingress-nginx/deploy/#aws

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
helm install ingress-nginx ingress-nginx/ingress-nginx -n laravel-app
```

Ingress Controller

Ingress NGINX

Using Ingress NGINX Controller

- Installation NGINX Ingress Controller using official HelmChart `ingress-nginx`.
- Validate K8S service for `ingress-nginx` exist.

```
kubectl --namespace laravel-app get services ingress-nginx-controller
```

Setup EKS Ingress Controller (NGINX, ALB) -1.2-


```
└─ helm install ingress-nginx ingress-nginx/ingress-nginx -n laravel-app

NAME: ingress-nginx
LAST DEPLOYED: Tue Nov  2 08:03:48 2021
NAMESPACE: laravel-app
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
The ingress-nginx controller has been installed.
It may take a few minutes for the LoadBalancer IP to be available.
You can watch the status by running 'kubectl --namespace laravel-app get services -o wide -w ingress-nginx-cor

An example Ingress that makes use of the controller:

  apiVersion: networking.k8s.io/v1
  kind: Ingress
  metadata:
    annotations:
      kubernetes.io/ingress.class: nginx
    name: example
    namespace: foo
  spec:
```

Ingress Controller

Ingress NGINX

Using Ingress NGINX Controller

- Installation NGINX Ingress Controller using official HelmChart `ingress-nginx`.
- Validate K8S service for `ingress-nginx` exist.

```
kubectl --namespace laravel-app get services ingress-nginx-controller
```

- Validate Service Ingress NGINX Controller

```
kubectl get services ingress-nginx-controller -n laravel-app
---
NAME                       TYPE           CLUSTER-IP      EXTERNAL-IP                                                PORT(S)                      AGE
ingress-nginx-controller   LoadBalancer   10.100.87.254   ab452***-***828.ap-southeast-1.elb.amazonaws.com   80:30186/TCP,443:30427/TCP   8m41s
```

Setup EKS Ingress Controller (NGINX, ALB) -2.1-

- Why we use Ingress ALB Controller ?
  - There's no default port exposed in K8S using port 80 (http) & 443 (https).
  - There's no SSL Termination that will "force" redirect from http to https.
  - As mandatory services while you're not using service mesh / service discovery (eg: Traefik / Istio).
  - Frontend Application deployment use this as mandatory instead Classic Load Balancer.

- IAM Policy & Cert Manager
  - Create custom IAM Policy for Ingress ALB
    Ref: https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.2.0/docs/install/iam_policy.json
  - Attach IAM Policy to EKS Cluster IAM (eg: attach role `eks-role-devopscorner_poc-nodes`)
  - Deploy Cert-Manager
    `kubectl apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v1.0.2/cert-manager.yaml -n laravel-app`

Ingress Controller

Ingress ALB

ALB (Application Load Balancer)

- Using customize IAM Policy for Ingress ALB.
- Attached to role EKS Cluster from terraform provisioned.
- Deploy Cert Manager `cert-manager.yaml`
- Deploy ALB Controller from yaml file
- References :
  https://kubernetes-sigs.github.io/aws-load-balancer-controller/v1.1/guide/controller/setup

Setup EKS Ingress Controller (NGINX, ALB) -2.2-

- Deploy Ingress ALB Controller
  - Download Manifest ALB Ingress Controller
    Ref: https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.9/docs/examples/alb-ingress-controller.yaml
  - Edit Manifest to add cluster name (--cluster-name=devCluster) to your EKS Cluster name.
  - Deploy RBAC Manifest
    ```
    kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.1.9/docs/examples/rbac-role.yaml
    ```
  - Deploy ALB Ingress Controller Manifest
    ```
    kubectl apply -f alb-ingress-controller.yaml
    ```
  - Verify Deployment Success
    ```
    kubectl logs -n kube-system $(kubectl get po -n kube-system | egrep -o "alb-ingress[a-zA-Z0-9-]+")
    ```

Ingress Controller

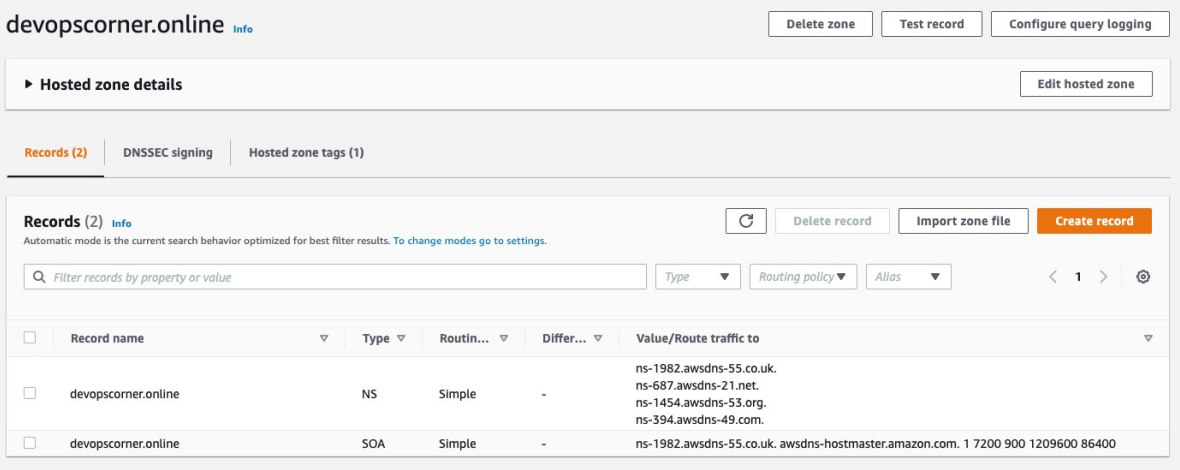
Ingress ALB

ALB (Application Load Balancer)

- Using customize IAM Policy for Ingress ALB.
- Attached to role EKS Cluster from terraform provisioned.
- Deploy Cert Manager `cert-manager.yaml`
- Deploy ALB Controller from yaml file
- References :
  https://kubernetes-sigs.github.io/aws-load-balancer-controller/v1.1/guide/controller/setup

Setup EKS DNS Route53 -1-

- Create Domain Zone in Amazon Route53



DNS Route

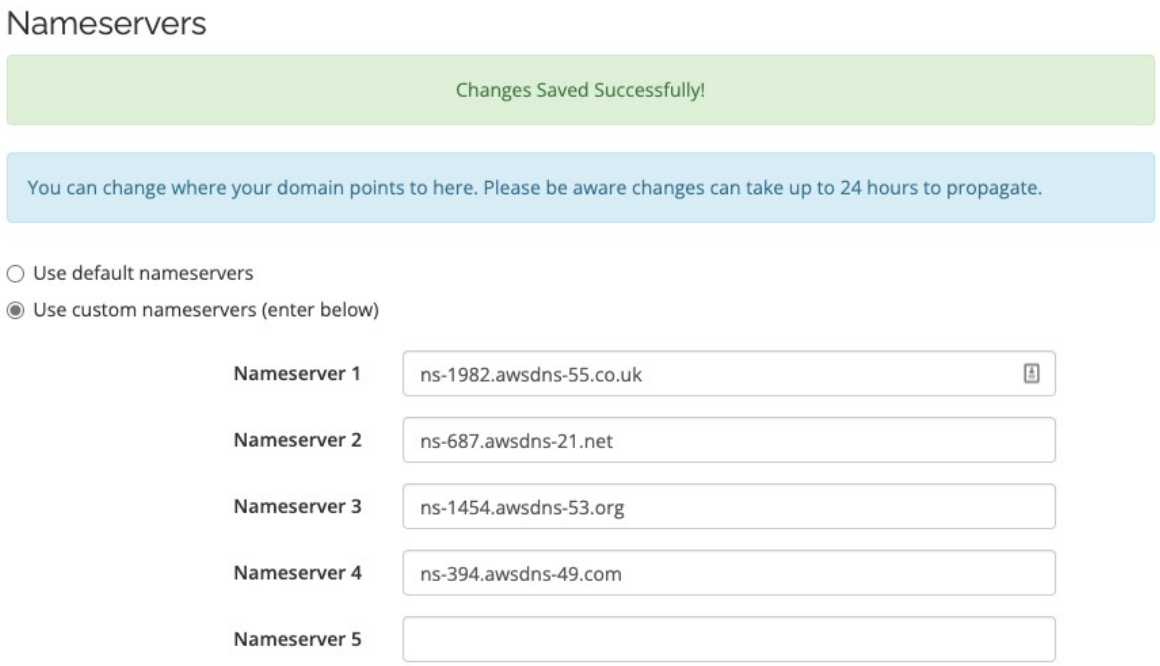
DNS Route53

Setup DNS inside Route53

- **References:**
  - Making Route 53 the DNS service for a domain that's in use - Amazon Route 53
  - Making Amazon Route 53 the DNS service for an existing domain - Amazon Route 53
  - Transferring registration for a domain to Amazon Route 53 - Amazon Route 53

Setup EKS DNS Route53 -2-

- Setup NS (Name Server) from Domain Panel



DNS Route

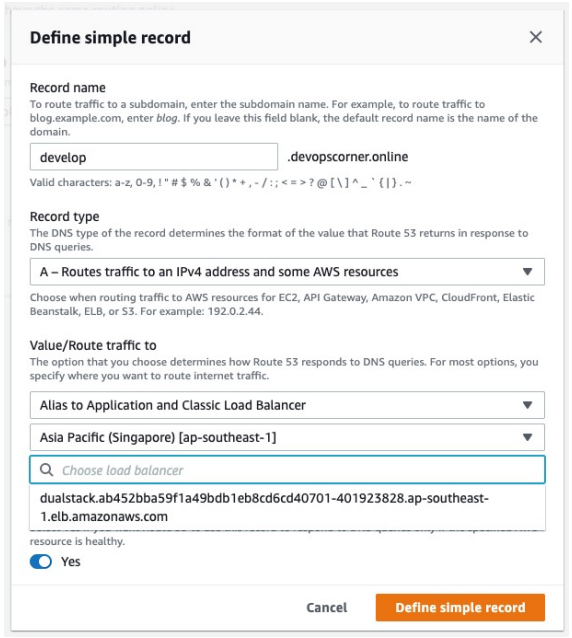
DNS Route53

Setup DNS inside Route53

- References:
  - Making Route 53 the DNS service for a domain that's in use - Amazon Route 53
  - Making Amazon Route 53 the DNS service for an existing domain - Amazon Route 53
  - Transferring registration for a domain to Amazon Route 53 - Amazon Route 53

Setup EKS DNS Route53 -3-

- Setup Subdomain Application from LB



- Validate Service Ingress NGINX Controller

```
kubectl get services ingress-nginx-controller -n laravel-app
---
NAME            : ingress-nginx-controller
TYPE            : LoadBalancer
CLUSTER-IP      : 10.100.87.254
EXTERNAL-IP     : ab452***-***828.ap-southeast-1.elb.amazonaws.com
PORT(S)         : 80:30186/TCP,443:30427/TCP
AGE             : 8m41s
```

DNS Route

DNS Route53

Setup DNS inside Route53

- References:
  - Making Route 53 the DNS service for a domain that's in use - Amazon Route 53
  - Making Amazon Route 53 the DNS service for an existing domain - Amazon Route 53
  - Transferring registration for a domain to Amazon Route 53 - Amazon Route 53

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

www.zebrax.id

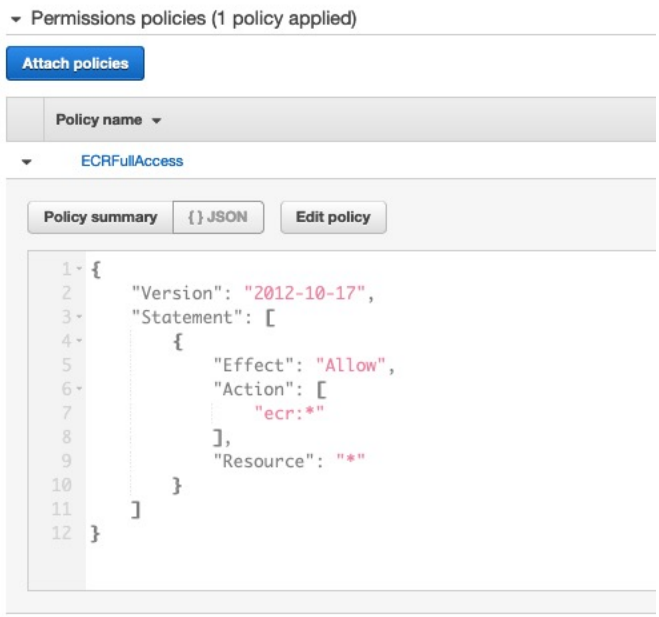
Contents

Build, Push & Pull Container Image

- Setup IAM Policy for ECR
- Login Docker Hub Account
- Create Dockerfile Application
- Build Dockerfile Image
- Push Container Image to Container Registry (ECR / Dockerhub)
- Pull Container Image from Container Registry (ECR / Dockerhub)
- Scan for Container Vulnerabilities

Setup IAM Policy for ECR

- IAM Custom Policy for ECR



IAM Policy

ECR IAM Policy

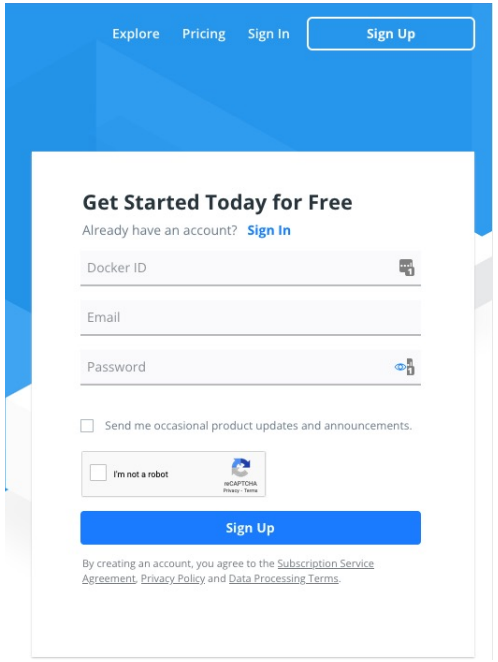
IAM Policy will be use as ECR authorization to push, pull or delete ECR container image

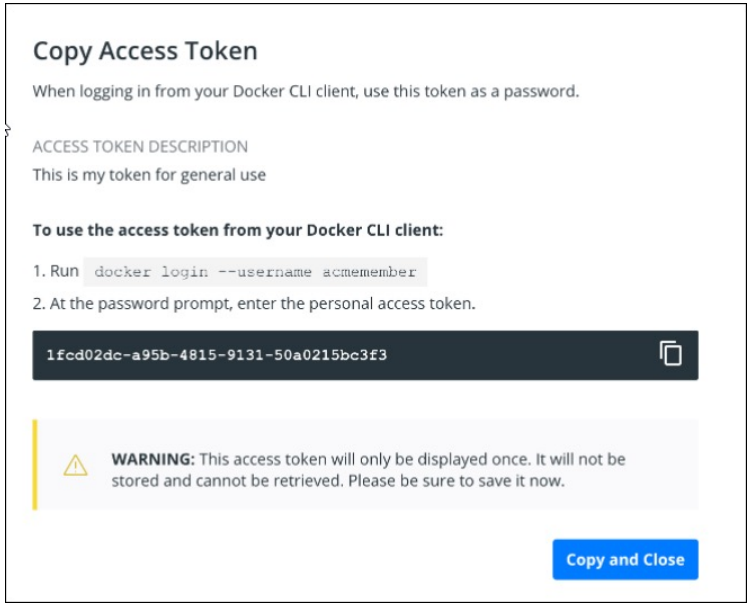
- Using AWS Console to create/update JSON IAM Policy for ECR.
- Use this IAM Custom Policy for accessing ECR from EC2 Instances.

- ECR Login (CLI)

```
aws ecr get-login-password --region ap-southeast-1 | docker login --username AWS -
-password-stdin [AWS_ACCOUNT_ID].dkr.ecr.ap-southeast-1.amazonaws.com
---
Login Succeeded
```

Login Docker Hub Account





- Docker Login (CLI)

```
└─ docker login
Authenticating with existing credentials...
Login did not succeed, error: Error response from daemon: Get "https://registry-1.docker.io/v2/": net
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, hea
Username (zeroc0d3):
Password:
Login Succeeded
```

Docker

Docker Hub

Logged into your Docker Hub account to get credential access (token)

- Register / login your email in Docker Hub (https://hub.docker.com).
- Use: `docker login` to get authorization token Docker Hub.
- Use: `cat ~/.docker/.token_seed` to get information token generated login.
- References:
  - https://docs.docker.com/docker-hub/access-tokens/

Create & Build Dockerfile

- PHPFpm Dockerfile

  Ref: Docker PHP-FPM 7.4 (github.com)

- Laravel Container

```
cd ~ && docker run --rm -v $(pwd):/app composer create-project --prefer-dist laravel/laravel laravel-kubernetes
```

- Docker Build

```
docker build [PATH_DOCKERFILE] -t [IMAGE_NAME]:[TAG_NAME]
# --- DockerHub ---
docker build /home/ubuntu/Laravel/Dockerfile -t zeroc0d3/laravel-kubernetes:latest

# --- ECR ---
docker build . -t [ECR_PATH]/laravel-kubernetes:latest
docker build . -t [AWS_ACCOUNT_ID].dkr.ecr.ap-southeast-1.amazonaws.com/devopscorner/laravel-kubernetes:latest
```

Dockerfile

Dockerfile Configuration

Install your dependencies libraries inside Container with Dockerfile. Use cascading (multistage) builder for reducing size of your Docker container images

- Register / login your email in Docker Hub
- Use: `docker build [PATH_DOCKERFILE] -t [IMAGE_NAME]:[TAG_NAME]` to build your container image from specific path of Dockerfile

Push & Pull Container Image from Container Registry

- Docker Push

```
docker push [IMAGE_NAME]:[TAG_NAME]
# --- DockerHub ---
docker push zeroc0d3/laravel-kubernetes:latest

# --- ECR ---
docker push [ECR_PATH]/laravel-kubernetes:latest
docker push [AWS_ACCOUNT_ID].dkr.ecr.ap-southeast-
1.amazonaws.com/devopscorner/laravel-kubernetes:latest
```

- Docker Pull

```
docker pull [IMAGE_NAME]:[TAG_NAME]
# --- DockerHub ---
docker pull zeroc0d3/laravel-kubernetes:latest

# --- ECR ---
docker pull [ECR_PATH]/laravel-kubernetes:latest
docker pull [AWS_ACCOUNT_ID].dkr.ecr.ap-southeast-
1.amazonaws.com/devopscorner/laravel-kubernetes:latest
```

Container Image

Push & Pull Container Image

Use ECR and/or Docker Hub as Container Registry for HelmChart Image Resources

- ECR Authorization using IAM Policy
- Docker Hub Authorization using Token from Login
- Use: `` `docker push [IMAGE_NAME]:[TAG_NAME]` `` to push build image from local to container registry
- Use: `` `docker pull [IMAGE_NAME]:[TAG_NAME]` `` to pull image from container registry to local

Scan for Container Vulnerabilities

- Scanning Container Image

  `trivy [IMAGE_NAME]:[TAG_NAME]`

```
zeroc0d3/laravel-kubernetes:latest (debian 11.1)
================================================
Total: 333 (UNKNOWN: 2, LOW: 47, MEDIUM: 176, HIGH: 96, CRITICAL: 12)

+-------------------------+------------------+----------+-------------------+---------------+--------------------------------------------------------------+
|         LIBRARY         | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |                            TITLE                             |
+-------------------------+------------------+----------+-------------------+---------------+--------------------------------------------------------------+
| apache2                 | CVE-2007-0086    | HIGH     | 2.4.51-1~deb11u1  |               | ** DISPUTED **  The Apache                                   |
|                         |                  |          |                   |               | HTTP Server, when accessed                                   |
|                         |                  |          |                   |               | through a...                                                 |
+                         +------------------+----------+                   +---------------+--------------------------------------------------------------+
|                         | CVE-2003-1307    | MEDIUM   |                   |               | ** DISPUTED **  The mod_php                                  |
|                         |                  |          |                   |               | module for the Apache HTTP                                   |
|                         |                  |          |                   |               | Server...                                                    |
+                         +------------------+          +                   +---------------+--------------------------------------------------------------+
|                         | CVE-2003-1580    |          |                   |               | The Apache HTTP Server 2.0.44,                               |
|                         |                  |          |                   |               | when DNS resolution is enabled                               |
|                         |                  |          |                   |               | for client...                                                |
+                         +------------------+          +                   +---------------+--------------------------------------------------------------+
|                         | CVE-2007-1743    |          |                   |               | suexec in Apache HTTP Server                                 |
|                         |                  |          |                   |               | (httpd) 2.2.3 does not verify                                |
|                         |                  |          |                   |               | combinations of...                                           |
+                         +------------------+          +                   +---------------+--------------------------------------------------------------+
|                         | CVE-2007-3303    |          |                   |               | Apache httpd 2.0.59 and 2.2.4,                               |
|                         |                  |          |                   |               | with the Prefork MPM module,                                 |
|                         |                  |          |                   |               | allows local...                                              |
+                         +------------------+----------+                   +---------------+--------------------------------------------------------------+
|                         | CVE-2001-1534    | LOW      |                   |               | mod_usertrack in Apache 1.3.11                               |
|                         |                  |          |                   |               | through 1.3.20 generates                                     |
|                         |                  |          |                   |               | session ID's using predictable                               |
|                         |                  |          |                   |               | information...                                               |
+                         +------------------+          +                   +---------------+--------------------------------------------------------------+
|                         | CVE-2003-1581    |          |                   |               | httpd: Injection of arbitrary                                |
|                         |                  |          |                   |               | text into log files when DNS                                 |
|                         |                  |          |                   |               | resolution is...                                             |
+                         +------------------+          +                   +---------------+--------------------------------------------------------------+
|                         | CVE-2008-0456    |          |                   |               | httpd: mod_negotiation CRLF                                  |
|                         |                  |          |                   |               | injection via untrusted file                                 |
|                         |                  |          |                   |               | names in directories with                                    |
|                         |                  |          |                   |               | MultiViews...                                                |
+-------------------------+------------------+----------+-------------------+---------------+--------------------------------------------------------------+
| apache2-bin             | CVE-2007-0086    | HIGH     |                   |               | ** DISPUTED **  The Apache                                   |
```

Scanning

Vulnerabilities Container Images

We use **trivy**, as binary to execute scanning container vulnerabilities inside container images

- Download binary from trivy repository
  https://github.com/aquasecurity/trivy
- Use: `trivy image [IMAGE_NAME]` to execute scanning vulnerabilities inside the container image
  - `trivy image zeroc0d3/laravel-kubernetes:latest`
  - `trivy image [ECR_PATH]/laravel-kubernetes:latest`

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

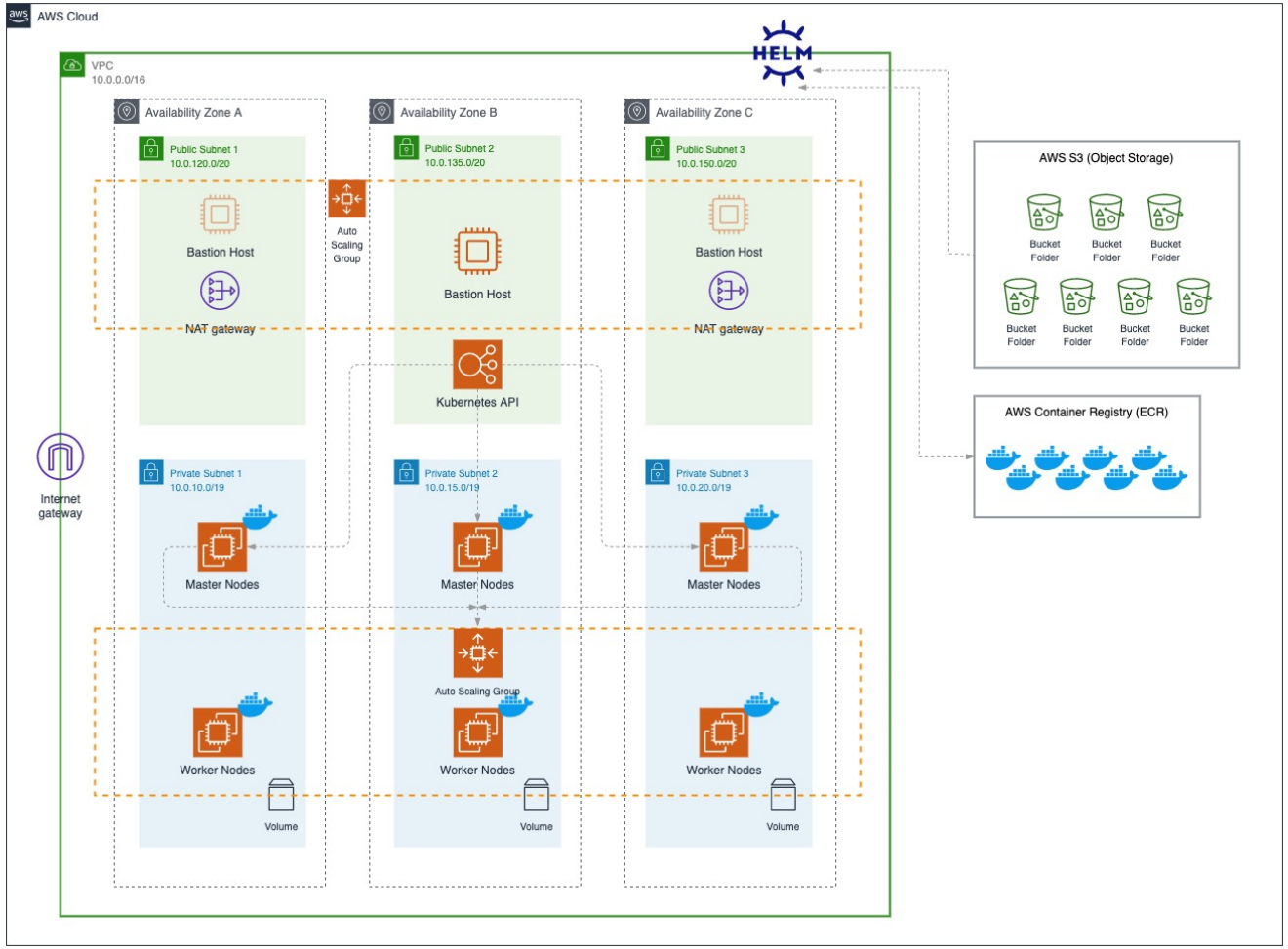
www.zebrax.id

Contents

HelmChart Template for K8S Resources

- Initiate HelmChart
- Create HelmChart Application
- Update Repository HelmChart
- Push HelmChart Template HelmChart to S3 (Object Storage)
- Update Template HelmChart to S3 (Object Storage)

Architecture Helm for EKS



Architecture

Helm CLI

Helm is an open-source package manager for K8S. Helm Charts help you define, install, and upgrade even the most complex Kubernetes application.

- What Helm can do ?
  - Manage Complexity
    Charts describe even the most complex apps, provide repeatable application installation, and serve as a single point of authority.
  - Easy Updates
    Take the pain out of updates with in-place upgrades and custom hooks.
  - Simple Sharing
    Charts are easy to version, share, and host on public or private servers.
  - Rollbacks
    Use helm rollback to roll back to an older version of a release with ease.

Create HelmChart Template

- HelmChart Template

```
helm create mychart && tree mychart

mychart
├── Chart.yaml
├── charts
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml

3 directories, 10 files
```

HelmChart

HelmChart Template

Configure pods, network, request & limit resources, ConfigMap, SecretRef, etc (K8S Resources) in single file yaml

- Initialize HelmChart with `helm init`
- Define Chart Repository in `Chart.yaml`
- Define Templates in `templates` folder
- Define Value HelmChart in `values.yaml`
- Push Template to Object Storage (S3)

Update HelmChart Repository

- List HelmChart Repository (Local)

```
helm repo ls
---
NAME          URL
stable        https://charts.helm.sh/stable
```

- Update HelmChart Repository

```
helm repo add [ChartName] [URL]
---
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo add laravel-kubernetes [S3_ENDPOINT]/laravel-app
```

- Export Helm Manifest

```
helm get manifest [HELM_RELEASE] -n [NAMESPACE] > [OUTPUT_YAML_FILE]
---
helm get manifest laravel-kubernetes -n laravel-app > ~/Desktop/laravel-app-
manifest.yml
```

HelmChart

HelmChart Template

Configure pods, network, request & limit resources, ConfigMap, SecretRef, etc (K8S Resources) in single file yaml

- Initialize HelmChart with `helm init`
- Define Chart Repository in `Chart.yaml`
- Define Templates in `templates` folder
- Define Value HelmChart in `values.yaml`
- Push Template to Object Storage (S3)

Push & Update HelmChart Template to S3 (Object Storage)

- Testing HelmChart Template

```
helm template ./laravel-kubernetes -f templates/laravel/values.yaml
helm template ./phpfpm -f templates/phpfpm/values.yaml
helm template ./svcroles -f templates/svcroles/values.yaml
```

- Create Compressed HelmChart Template

```
helm package laravel-kubernetes
helm package phpfpm
helm package svcroles
```

- Push Chart into S3 Bucket

```
## Install S3 Plugins
helm plugin install https://github.com/hypnoglow/helm-s3.git

helm s3 push laravel-kubernetes-0.1.0.tgz devopscorner --force
helm s3 push phpfpm-0.1.0 devopscorner --force
helm s3 push svcroles-0.1.0.tgz devopscorner --force
```

HelmChart

HelmChart Template

Configure pods, network, request & limit resources, ConfigMap, SecretRef, etc (K8S Resources) in single file yaml

- Initialize HelmChart with `helm init`
- Define Chart Repository in `Chart.yaml`
- Define Templates in `templates` folder
- Define Value HelmChart in `values.yaml`
- Push Template to Object Storage (S3)

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

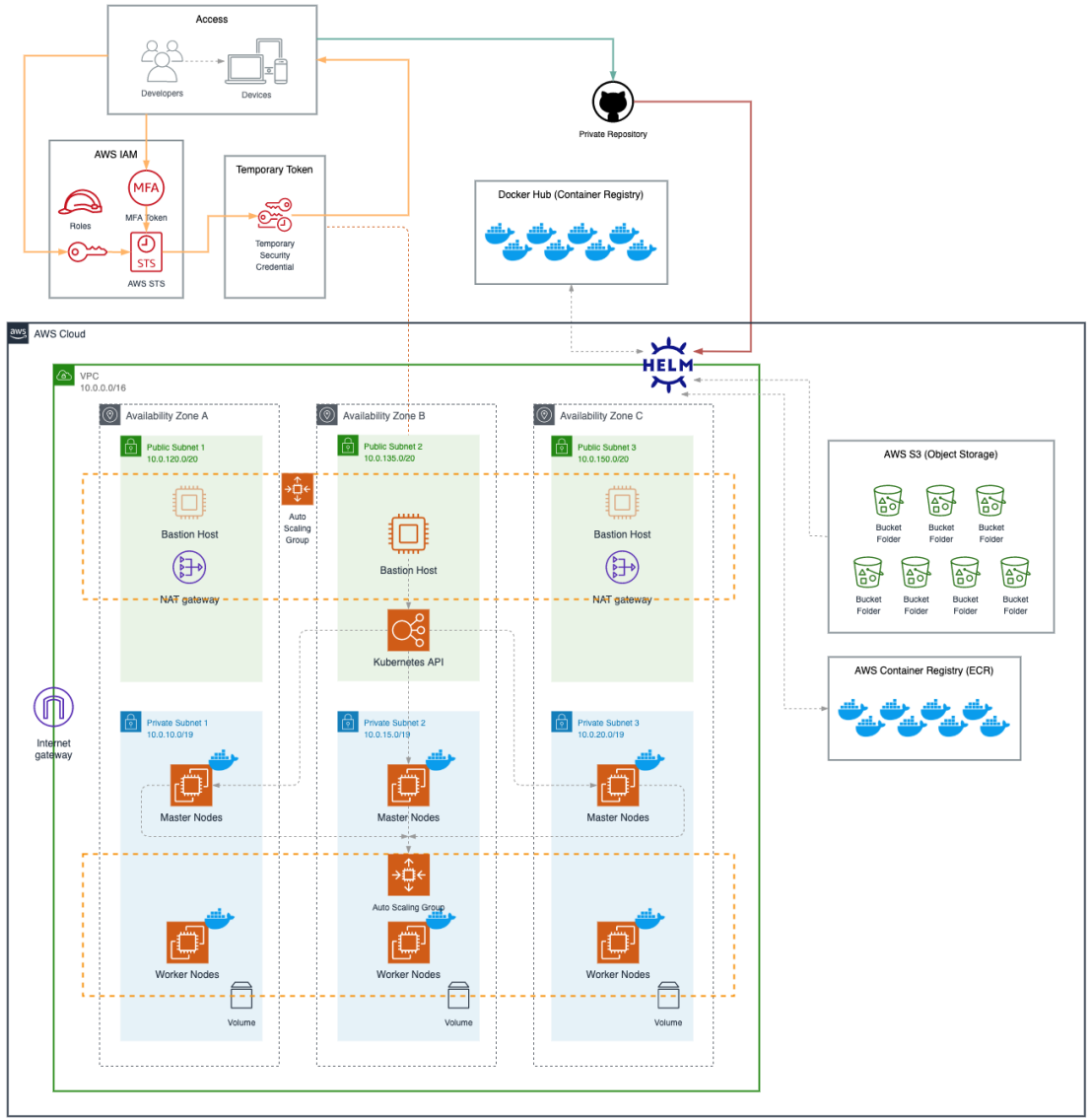
www.zebrax.id

Contents

Deploy HelmChart to EKS

- Deploy HelmChart using Container Registry (ECR / Docker Hub)
- Deploy HelmChart with / without Bastion
- Deploy HelmChart with CI/CD Tools

Architecture Deployment



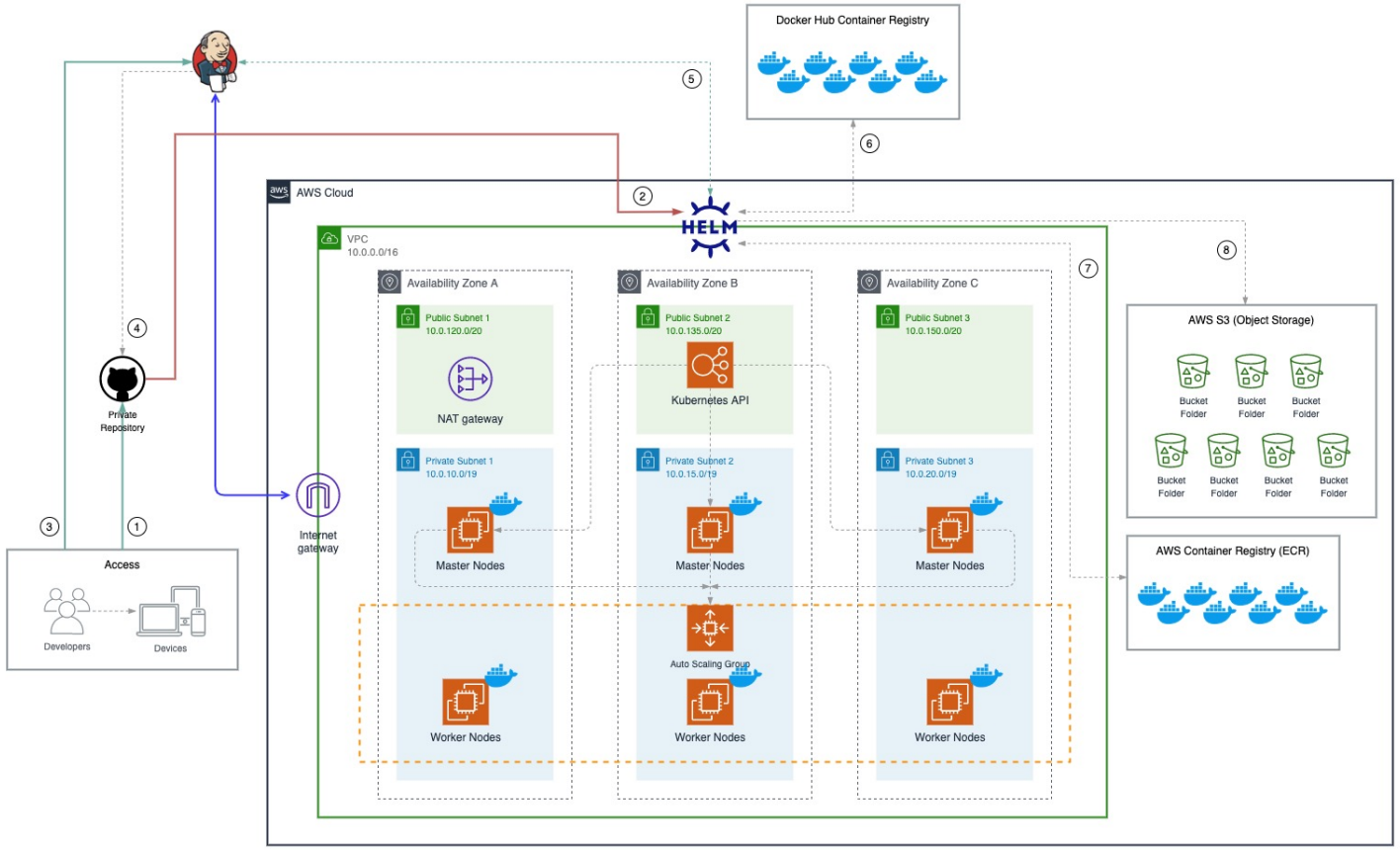
Architecture

HelmChart CLI

Deploy HelmChart to EKS Cluster using secure connection inside Bastion EC2.

- Commit & push source code to Private Repository (including HelmChart configuration)
- Access secure connection ssh to bastion EC2 for deployment Helm
- Build, Push & Pull Image from ECR & Docker Hub
- Authorize EKS cluster using IAM Policy
- Deploy Helm from bastion instance EC2

CI/CD Pipeline Deployment



Jenkins

Jenkins Pipeline

HelmChart pipeline deployment configured inside Jenkinsfile (.jenkinsfile)

- Commit & push source code to Private Repository (including HelmChart configuration)
- Access Jenkins to trigger job deployment
- Jenkins will execute:
  - Pull source code from Private Repository
  - Build container image from source code
  - Push container image to ECR / Docker Hub
  - Pull container image from latest version or commit hash
  - Helm deployment

Prerequirements Environment -1-

- Check HelmChart Repository

```
helm repo ls
---
NAME.    URL
stable   https://charts.helm.sh/stable
```

- Add 3rd Party Repository (Bitnami)

```
helm repo add bitnami https://charts.bitnami.com/bitnami
---
"bitnami" has been added to your repositories
```

- Validate Updated HelmChart Repository

```
helm repo ls
---
NAME.    URL
stable   https://charts.helm.sh/stable
bitnami  https://charts.bitnami.com/bitnami
```

- Deploy MariaDB HelmChart

```
helm install mariadb \
--set auth.rootPassword=secretpassword,auth.database=laravel_db \
bitnami/mariadb -n laravel-app

--or—

## Ref: https://github.com/bitnami/charts/blob/master/bitnami/mariadb/values.yaml

helm install mariadb -f values.yaml bitnami/mariadb -n laravel-app
```

Helm CLI

Helm CLI Deployment

Deploy HelmChart with Helm CLI using definition resources from Helm Templates

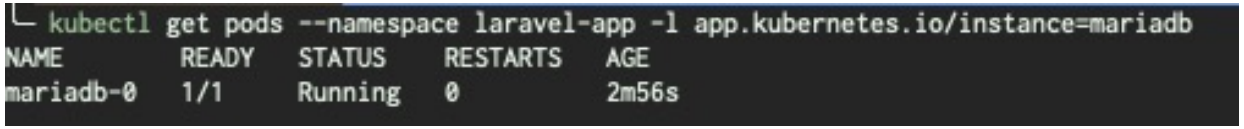
- Check / validate your HelmChart repository exist and accessible.
- Add Bitnami Repository as 3rd party HelmChart.
- Deploy HelmChart Repository 3rd party using Helm CLI or you can have customize config HelmChart value inside `values.yaml` file
- Make sure your HelmChart template already set with correct value inside `values.yaml` file.
- Upgrade value HelmChart from cli:

```
helm upgrade --namespace laravel-app laravel-mariadb
bitnami/mariadb --set
auth.rootPassword=[YOUR_ROOT_PASSWORD]
```

Prerequirements Environment -2-

- Validate MariaDB HelmChart Deployed

```
kubectl get pods --namespace laravel-app -l app.kubernetes.io/instance=mariadb
---
NAME                READY   STATUS    RESTARTS   AGE
mariadb-0           1/1     Running   0          2m56s
```



- Validate MariaDB Connection Pods

```
## Run pods as client

kubectl run laravel-mariadb-client --rm --tty -i --restart='Never' --image
docker.io/bitnami/mariadb:10.5.12-debian-10-r68 --namespace laravel-app --command
-- bash

--or--

exec kubectl exec -i -t -n laravel-app mariadb-0 -c mariadb "--" sh -c "clear;
(bash || ash || sh)"


## Connection to primary service (read/write)

mysql -h mariadb.laravel-app.svc.cluster.local -uroot -p laravel_db
```

Helm CLI

Helm CLI Deployment

Deploy HelmChart with Helm CLI using definition resources from Helm Templates

- Check / validate your HelmChart repository exist and accessible.
- Add Bitnami Repository as 3rd party HelmChart.
- Deploy HelmChart Repository 3rd party using Helm CLI or you can have customize config HelmChart value inside `values.yaml` file
- Make sure your HelmChart template already set with correct value inside `values.yaml` file.
- Upgrade value HelmChart from cli:

```
helm upgrade --namespace laravel-app laravel-mariadb
bitnami/mariadb --set
auth.rootPassword=[YOUR_ROOT_PASSWORD]
```

Prerequirements Environment -3-

- Validate MariaDB Connection Pods

```
## Run pods as client

kubectl run laravel-mariadb-client --rm --tty -i --restart='Never' --image
docker.io/bitnami/mariadb:10.5.12-debian-10-r68 --namespace laravel-app --command
-- bash

--or--

exec kubectl exec -i -t -n laravel-app mariadb-0 -c mariadb "--" sh -c "clear;
(bash || ash || sh)"


## Connection to primary service (read/write)

mysql -h mariadb.laravel-app.svc.cluster.local -uroot -p laravel_db
```

```
└─ exec kubectl exec -i -t -n laravel-app mariadb-0 -c mariadb "--" sh -c "clear; (bash || ash || sh)"

sh: 1: clear: not found
I have no name!@mariadb-0:/$ mysql -h mariadb.laravel-app.svc.cluster.local -uroot -p laravel_db
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 225
Server version: 10.5.12-MariaDB Source distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [laravel_db]>
```

Helm CLI

Helm CLI Deployment

Deploy HelmChart with Helm CLI using definition resources from Helm Templates

- Check / validate your HelmChart repository exist and accessible.
- Add Bitnami Repository as 3rd party HelmChart.
- Deploy HelmChart Repository 3rd party using Helm CLI or you can have customize config HelmChart value inside `values.yaml` file
- Make sure your HelmChart template already set with correct value inside `values.yaml` file.
- Upgrade value HelmChart from cli:

```
helm upgrade --namespace laravel-app laravel-mariadb
bitnami/mariadb --set
auth.rootPassword=[YOUR_ROOT_PASSWORD]
```

Deploy HelmChart Laravel -1-

- Configuration Environment `helm/secrets.yaml`

```
## Migrate Dotenv (.env) to Secret

php:
  envVars:
    - name: APP_KEY
      value: "your_laravel_app_key"
    - name: DB_DATABASE
      value: your_database_name
    - name: DB_USERNAME
      value: your_database_user
    - name: DB_PASSWORD
      value: "your_database_password”
    - name: DB_HOST
      value: "your_host_database"
```

- Configuration Helm Value `helm/values.yaml`

```
php:
  repository: zeroc0d3/laravel-kubernetes
  pullPolicy: IfNotPresent
  tag: "latest"
  fpmEnabled: false
  envVars:
    - name: APP_ENV
      value: production
    - name: APP_DEBUG
      value: false
    - name: DB_PORT
      value: 3306
```

Laravel Chart

Laravel HelmChart

Deploy Laravel with Customize HelmChart

- Migrate dotenv (.env) file to secret config K8S `secrets.yaml`.
- Create Helm Value for Laravel Kubernetes
- Deploy Helm Value Laravel Kubernetes
- Validate Deployment Helm Release (Laravel Kubernetes)
- Update Helm Value to set NGINX Ingress & Domain
- Redeploy Helm Value Laravel Kubernetes
- Setup SSL TLS for Domain (force redirect https)

Deploy HelmChart Laravel -2-

- Deploy Laravel-App Helm

```
helm install laravel-kubernetes -f values.yaml -f secrets.yaml stable/lamp -n laravel-app
```

```
└─ helm install laravel-kubernetes -f values.yaml -f secrets.yaml stable/lamp -n laravel-app

WARNING: This chart is deprecated
NAME: laravel-kubernetes
LAST DEPLOYED: Tue Nov  2 17:31:37 2021
NAMESPACE: laravel-app
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
INIT:
      Please wait for all init containers to finish before connecting to
      the charts services. This might take a few minutes depending on their
      tasks.

LOADBALANCER:
      Please wait until the service has been routed to an IP address.
      You can watch the status of by running 'kubectl get svc -w laravel-kubernetes-lamp'

1. You can now connect to the following services:
      export CHARTIP=$(kubectl get svc laravel-kubernetes-lamp --output=jsonpath={.status.loadBalancer.ingress..ip})

      Main Site:
        http://$CHARTIP
```

- Validate Deployment Laravel App

```
kubectl get pods -n laravel-app
```

Laravel Chart

Laravel HelmChart

Deploy Laravel with Customize HelmChart

- Migrate dotenv (.env) file to secret config K8S `secrets.yaml`.
- Create Helm Value for Laravel Kubernetes
- Deploy Helm Value Laravel Kubernetes
- Validate Deployment Helm Release (Laravel Kubernetes)
- Update Helm Value to set NGINX Ingress & Domain
- Redeploy Helm Value Laravel Kubernetes
- Setup SSL TLS for Domain (force redirect https)

Deploy HelmChart Laravel -3-

- Update Values `helm/values.yaml`

```
php:
  repository: zeroc0d3/laravel-kubernetes
  pullPolicy: IfNotPresent
  tag: "latest"
  fpmEnabled: false
  envVars:
    - name: APP_ENV
      value: production
    - name: APP_DEBUG
      value: false
    - name: DB_PORT
      value: 3306

autoscaling:
  enabled: true
  minReplicas: 1
  maxReplicas: 2
  targetCPUUtilizationPercentage: 80
  targetMemoryUtilizationPercentage: 80

nodeSelector:
  enabled: true
  select:
    node: "laravel"

ingress:
  enabled: false
  annotations: {}
  hosts:
    - host: ""
      paths: []
  tls: []
```

Laravel Chart

Laravel HelmChart

Deploy Laravel with Customize HelmChart

- Migrate dotenv (.env) file to secret config K8S `secrets.yaml`.
- Create Helm Value for Laravel Kubernetes
- Deploy Helm Value Laravel Kubernetes
- Validate Deployment Helm Release (Laravel Kubernetes)
- Update Helm Value to set NGINX Ingress & Domain
- Redeploy Helm Value Laravel Kubernetes
- Setup SSL TLS for Domain (force redirect https)

Deploy HelmChart Laravel -4-

- Redeploy Laravel-App Helm (Update)

```
helm upgrade laravel-kubernetes -f helm/values.yaml -f helm/secrets-prod.yaml
stable/lamp -n laravel-app
```

```
└─ helm upgrade laravel-kubernetes -f values.yaml -f secrets-prod.yaml stable/lamp -n laravel-app
WARNING: This chart is deprecated
Release "laravel-kubernetes" has been upgraded. Happy Helming!
NAME: laravel-kubernetes
LAST DEPLOYED: Wed Nov  3 14:17:11 2021
NAMESPACE: laravel-app
STATUS: deployed
REVISION: 31
TEST SUITE: None
NOTES:
INIT:
      Please wait for all init containers to finish before connecting to
      the charts services. This might take a few minutes depending on their
      tasks.

LOADBALANCER:
      Please wait until the service has been routed to an IP address.
      You can watch the status of by running 'kubectl get svc -w laravel-kubernetes-lamp'

1. You can now connect to the following services:
      export CHARTIP=$(kubectl get svc laravel-kubernetes-lamp --output=jsonpath={.status.loadBalancer.ingress..ip})

      Main Site:
        http://$CHARTIP
```

Laravel Chart

Laravel HelmChart

Deploy Laravel with Customize HelmChart

- Migrate dotenv (.env) file to secret config K8S `secrets.yaml`.
- Create Helm Value for Laravel Kubernetes
- Deploy Helm Value Laravel Kubernetes
- Validate Deployment Helm Release (Laravel Kubernetes)
- Update Helm Value to set NGINX Ingress & Domain
- Redeploy Helm Value Laravel Kubernetes
- Setup SSL TLS for Domain (force redirect https)

PHPID COMMUNITY

AWS community builder

Organizing Partner :

ZEBRAX

www.zebrax.id

Contents

ZEBRA X

PHPID COMMUNITY

AWS community builder

Thank You

PT ZEBRA CROSS TEKNOLOGI

INDY Bintaro Office Park, Building A, 5th floor
Tangerang Selatan 15224

Contact us: inquiry@zebrax.id

www.zebrax.id

Data are about understanding…

… to understand your opportunity is to understand your data

ZEBRA

Enabling Industry 4.0